NUCLEAR SCIENCE AND ENGINEERING · VOLUME 182 · 1–12 · JANUARY 2016

Extremely Accurate Sequential Verification of RELAP5-3D

G. L. Mesina,[a]* D. L. Aumiller,[b] and F. X. Buschman[b]

[a]*Idaho National Laboratory, 2525 Fremont Avenue, Idaho Falls, Idaho 83402*
[b]*Bettis Atomic Power Laboratory, 814 Pittsburgh McKeesport Boulevard, West Mifflin, Pennsylvania 15122*

Received December 10, 2014
Accepted for Publication June 1, 2015
http://dx.doi.org/10.13182/NSE14-151

Abstract — *Large computer programs like RELAP5-3D solve complex systems of governing, closure, and special process equations to model the underlying physics of thermal-hydraulic systems and include specialized physics for the modeling of nuclear power plants. Further, these programs incorporate other mechanisms for selecting optional code physics, input, output, data management, user interaction, and post-processing. Before being released to users, software quality assurance requires verification and validation. RELAP5-3D verification and validation are focused toward nuclear power plant applications. Verification ensures that the program is built right by checking that it meets its design specifications, comparing coding algorithms to equations, comparing calculations against analytical solutions, and the method of manufactured solutions.*

Sequential verification performs these comparisons initially, but thereafter only compares code calculations between consecutive code versions to demonstrate that no unintended changes have been introduced. An automated, highly accurate sequential verification method, based on previous work by Aumiller, has been developed for RELAP5-3D. It provides the ability to test that no unintended consequences result from code development. Moreover, it provides the means to test the following code capabilities: repeated time-step advancement, runs continued from a restart file, and performance of coupled analyses using the R5EXEC executive program. Analyses of the adequacy of the checks used in these comparisons are provided.

Keywords — *RELAP5-3D, verification, governing equations*.

Note — *Some figures may be in color only in the electronic version.*

I. INTRODUCTION

Before each product release, RELAP5-3D (Ref. 1) undergoes verification and validation as defined in IEEE-STD-610 (Ref. 2). Validation is the process of evaluating a system or component (software) during or at the end of the development process to determine whether it satisfies specified requirements, i.e., that it will fulfill its intended use. For RELAP5-3D, the special form of validation testing called developmental assessment[3] is performed. Verification evaluates a system or component (software) to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. Verification compares coding against its documented algorithms and equations and also compares its calculations against analytical solutions and the method of manufactured solutions.

Sequential verification[4] checks coding against specifications (documented algorithms and equations) only when originally implemented and then applies regression testing[5] to compare code calculations from consecutive updates or versions on a set of test cases to ensure that the performance does not change. If unexplained differences are not detected, the new release is sequentially verified. Explainable differences include those induced by error corrections, code improvements, and changes in the operating environment such as compiler and math library updates. Unexplained differences are treated as coding

*E-mail: george.mesina@inl.gov

errors and must be identified and corrected for sequential verification. During its life cycle, RELAP5-3D has employed sequential verification, expanding its test set with each new feature and capability.

As new features are incorporated into the code, pertinent test cases are added. Growth of the test set led to the development of the diffem utility[6,7] to automate character-by-character comparison of printed output files from two different versions of RELAP5-3D and report every difference. However, because printed output files record as few as five decimal places, this has proven insufficient to guarantee that development did not cause unintended changes in calculations. Moreover, analysis shows that some code features and capabilities, including restart and backup, were not checked by existing test suites.

Therefore, state-of-the-art verification techniques[4] were adapted and extended to the RELAP5-3D code to provide both sufficiently accurate detection of differences and comprehensive coverage of code features and capabilities.

II. DETECTION: THE VERIFICATION FILE

One means to verify the code sequentially is to write all data in RELAP5-3D memory to a binary file (for perfect precision). Comparing the calculations from two RELAP5-3D runs of an identical input file would then reveal every difference, or prove that none occurred. This is impractical and costly because

A. every new variable created must be added to the write statements

B. binary files with different variables (e.g., different code versions) are difficult to compare

C. writing every variable can greatly increase code run time

D. modeling detail and number of writes control the size of the file, so it can grow without bound

Despite powerful compression techniques, issue D can overfill disk space. The principal ways to reduce size are the restriction of output frequency and the reduction of the amount of data written. Toward this end, three major categories of data are identified:

1. primary variables from the governing equations for thermal hydraulics, heat transfer, neutronics, controls, and trips, e.g., fluid internal energy, heat structure temperature, and neutron flux (see Table I)
2. secondary variables derived from primary and other secondary variables and used to construct the discrete equations solved for primary variables, e.g., viscosity, heat capacity, and power
3. output-only variables that do not feed back into primary or secondary variables, e.g., mass residual

TABLE I

RELAP5-3D Primary Variables

Quantity	In Manual	On File
Pressure	p	P
Liquid internal energy	u_f	Uf
Gas internal energy	u_g	Ug
Void fraction of gas	α_g	VOIDg
Noncondensable quality	X_a	QUALa
Liquid velocity	V_f	Vf
Gas velocity	V_g	Vg
Heat structure temperature	T	Temp
Neutron flux	φ	Flux
Time step sum	$\Delta t, \Delta t_{\text{kin}}$	dtsum
Trips	T_r	Trips
Control system value	Y	Cntrl

If two runs of the same input model produce differences in a primary variable, variables in categories 2 and 3 that are derived from it must also differ. If a secondary variable differs, it affects the equations of at least one primary variable and therefore its value when the equation is solved. Thus, it is unnecessary to write category 2 variables, which constitute the bulk of memory, to the verification file. However, category 3 variables do not affect category 1. Therefore, writing every category 1 and category 3 variable on the file is necessary and sufficient for detecting every difference and eliminates issue C.

Since the choice of primary variables is rarely changed or modified, writing only category 1 variables mitigates issues A and B, but it allows a few differences to escape detection. Issue D is overcome by calculating the L_1-norm of each primary variable array in quadruple precision for extreme accuracy, which allows ASCII output in both scientific notation and 32-place hexadecimal that represents all bits of the sum. Moreover, the file size is restricted; once the size limit of 1 MB is reached, output is suspended until the final time step.

II.A. Selection of Primary Variables

Primary variables are identified by the field equations solved by RELAP5-3D. Equations (3.1-2), (3.1-3), (3.1-6), (3.1-7), (3.1-11), (3.1-12), and (3.1-39) of Ref. 1 give the governing equations for thermal hydraulics as conserva-

When some L_1-norm differs between consecutive versions for a particular input file, these sums also serve as an initial debugging aid. The exact time step when the first difference occurs can be found by rerunning with verification file dumps made on every time step. On the first time step with differences, the solution sequence helps provide clues: trips, heat transfer, hydrodynamics, kinetics, and controls. If differences occur in one of these phenomenological areas and nowhere else, the related subroutines and secondary variables of that area are examined first for the source. If differences occur in two or more groups, the first area in time-step calling order is examined for the source of differences. For example, if the primary variable or variables that differ occur in the thermal hydraulics group (first ten sums in Fig. 1), this indicates that thermal hydraulics subroutines and thermal hydraulics secondary variables should be examined first for the source. This subdivides the secondary and output variables into groups, by physics, based on their association with primary variable groups through the related FORTRAN-90 modules, thereby aiding the debugging effort.

This approach to initial debugging also applies to output variable sums in Fig. 1. If a reduction or repetition count differs, the thermal hydraulics area is examined first. If the error or time sum differs, the individual error measures and times are examined to determine which primary group is associated with the difference.

III. COVERAGE: THE VERIFICATION TEST SUITE

Traditional coverage determines the percentage of lines of code exercised by a test suite. However, in RELAP5-3D verification, coverage is measured by the number of code features used in the modeling of nuclear power plants that are tested. Since so many features exist, a judgment has been made to exclude initially those not used in the developmental assessment[3] and other standard test suites as well as those rarely used for nuclear power models. Therefore, current coverage is not 100% of all code features, but 194 important ones are verified. Code features tested by the suite are lumped into the following categories[8]:

1. hydrodynamic components: pipes, separators, etc.
2. control volume flags: thermal stratification, mixture level, etc.
3. additional wall friction options: shape factor, viscosity ratio, etc.
4. junction flags: jet junction, countercurrent flow limiting, etc.
5. junction form loss: constant, abrupt area change, etc.
6. heat structure geometry type: rectangular, cylindrical, spherical
7. heat structure boundary conditions: adiabatic, convective, etc.
8. heat source options: radial factor shape, table, etc.
9. material properties: built-in, user input (functions and tables)
10. control functions: arithmetic operations, controllers, etc.
11. trips: logical or variables
12. general tables: power, temperature, etc.
13. reactor kinetics: point, nodal
14. decay heat: No decay heat, ANS/ANSI standard options
15. equation solvers: BPLU, PGMRES, LSOR, Krylov, etc.
16. time-step integration schemes: semi-implicit, nearly implicit.

A number of other code features that do not fit these categories, such as the use of noncondensables and cases with or without boron tracking, are also included. Because user choices affect the way the code operates, certain developmental (card-1) options are also included as features. Among excluded features are many of the available working fluids, numerous developmental (card-1) options, special output control, and debug options.

IV. FEATURES TESTS MATRIX

The verification test suite[6,7] comprises 43 input decks having 125 separate input cases that test 194 code features. Typically, decks with multiple cases specifically vary one feature, e.g., the heat transfer mode. Features tested in one kind of hydrodynamic component, e.g., a pipe, need not be tested in another, e.g., valves. A subset of input decks includes problems with analytical or well-known solutions, e.g., control variable functions with exact mathematical solutions.

The features tests matrix,[4,6,7] with over 200 rows and 46 columns, is broken into six sub-tables; one is shown in Table II. Column 1 lists the features tested, and the input decks are across the top. A mark in column 2 indicates that some input deck tests the feature. Marks in columns 3 and 4 indicate restart and backup testing, which are discussed in Secs. V.B and V.C, respectively. A mark in any remaining