INTEGRATION OF THE MONK MONTE CARLO NEUTRONICS CODE IN THE NURESIM PLATFORM

F. Tantillo, S. D. Richards, A. J. Smethurst and D. J. Long
ANSWERS Software Service, Wood
Kings Point House, Queen Mother Square, Poundbury, Dorchester, United Kingdom, DT1 3BW

simon.richards@woodplc.com

ABSTRACT

The MONK® Monte Carlo neutronics code has been integrated into the NURESIM nuclear reactor simulation platform and coupled to the subchannel thermal hydraulics code SUBCHANFLOW. A PWR pincell calculation is used to demonstrate the coupled code system, results of which agree well with with independent calculations carried out with WIMS coupled with its own sub-channel module ARTHUR.

KEYWORDS: Monte Carlo, neutronics, thermal hydraulic, coupling

1. INTRODUCTION

The Horizon 2020 McSAFE project is a collaborative effort involving 12 institutions from 7 EU countries to provide reliable and efficient numerical tools for the simulation of light water reactors (LWRs). One of the main objectives of the project is to develop valuable numerical tools for realistic core design, safety analysis and industry-like applications of LWRs, including optimal coupling of Monte Carlo (MC) codes to thermal-hydraulic (TH) solvers, and time-dependent Monte Carlo methods. These tools are being developed in the framework of the NURESIM [1] NUclear REactor SIMulation platform, in order to consolidate and further its use, and also to benefit from its innovative coupling approaches. NURESIM provides, within the SALOME [2] open source integration platform, an integrated set of European software at the state of the art or beyond.

In order to extend the range of numerical tools available in the NURESIM platform the UK Monte Carlo code MONK® [3,4,5] has been integrated with the aim of coupling it to thermal-hydraulic solvers. MONK is an advanced Monte Carlo neutronics code for the solution of criticality safety and reactor physics problems with a proven track record of application to the whole of the nuclear fuel cycle. It is a well-established, regulator-recognized, commercial code which is routinely applied to the solution of industrial-strength problems and in support of nuclear safety cases. The degree to which a commercial code such as MONK can be integrated in the NURESIM platform can be used to assess the maturity of the platform and its suitability for deployment to industrial users.

2. NEUTRONICS AND THERMAL HYDRAULICS COUPLING

2.1. General Description of the MONK Monte Carlo Code

MONK is a powerful 3D Monte Carlo code for nuclear criticality safety and reactor physics analyses, forming part of the ANSWERS® codes suite. ANSWERS codes are widely used in over thirty countries around the world, and on a range of reactor types including: AGR, BWR, CANDU, MAGNOX, RBMK, PBMR, PWR, VVER and many experimental reactors. The codes have already been applied to some of the future reactor technologies (e.g. high temperature and fast breeder reactors in the Generation IV programme) that are being developed not only for electricity generation but additionally for other applications. MONK's advanced geometry modelling and detailed continuous energy collision treatment provides realistic 3D models for an accurate simulation of neutronic behaviour. MONK's superhistory powering algorithm [6] provides robust and reliable estimates of the neutron multiplication factor and other parameters of interest, even for highly decoupled systems.

MONK has both continuous energy and multigroup capabilities, and is supplied with nuclear data libraries in both the BINGO continuous energy format and the WIMS 172 group format, based on evaluations including recent JEFF, ENDF and CENDL libraries. The BINGO continuous energy collision processor and data libraries support run-time Doppler broadening for accurate temperature representation, which is important when modelling thermal feedback. An internally-coupled depletion solver allows MONK to calculate microscopic burnup using either continuous energy or multigroup data.

MONK features an easy to use, flexible and powerful geometry package comprising two components: Fractal Geometry which allows detailed geometries to be constructed hierarchically from simple components; and Hole Geometry (using Woodcock tracking [7]) which is used extensively in MONK to provide a wide range of more complicated fine geometric details and to expedite the specification of commonly-occurring replicating items. The two components can be mixed freely to define the required geometry in the most efficient way. Additionally MONK is able to import the model geometry directly from CAD files and track neutrons directly in the CAD geometry without conversion or approximation.

The Unified Tally module in MONK provides a flexible means for defining tally bodies and meshes in a way that is decoupled from the underlying model geometry. Any number of independent scoring bodies may be used, they may overlap, and each may have its own defined scoring energy group scheme. Optionally the events scored within each mesh may be further broken down by material. Many of the recent features of MONK are based on Unified Tally meshes, including the Shannon entropy module, the fission matrix module, the mesh-based burnup capability, and the AT-in-UT module which tallies reaction rates in a mesh, optionally subdivided by material and nuclide. The mesh-based burn-up option uses Unified Tally meshes to define both a burnup (BU) mesh which automates the process of producing unique materials in each depleteable zone, and a thermal hydraulics (TH) mesh for externally coupling to TH codes. Fission heating powers can be computed in the TH mesh, and modified material temperatures and densities can be returned to MONK from a TH code using the same mesh.

The current QA-release version of MONK is MONK10B which was released in July 2017. This is

available for both Windows and Linux platforms. MONK supports MPI parallelization using Open MPI (Open MPI version 1.6.5 in MONK10B) using a master-slave algorithm. Future releases are also planned to support hybrid MPI / OpenMP parallelization or some other form of shared memory parallelism.

MONK has a proven track record of application to the whole of the nuclear fuel cycle and is well established in the UK criticality community as the *de facto* standard Monte Carlo criticality code. It has been in continuous development and use since the 1960s and is developed, distributed, licensed and actively supported by the ANSWERS Software Service [8], part of Wood.

2.2. Software Description and Development Environment

MONK is developed in Fortran. It is part of a suite of ANSWERS codes known collectively as the MCANO codes, which include MONK, MCBEND (a Monte Carlo general radiation transport code), RANKERN (a point kernel gamma shielding code), CACTUS (a method-of-characteristics module for the WIMS modular reactor physics code suite), and VRFORT (the ray-tracing engine used for geometry visualization in the ANSWERS Visual Workshop integrated development environment). All of the MCANO codes share a common code base and Subversion (SVN) repository, and a common build system which selects the required source files from the common repository depending on which code it is building. The size of the MCANO and MONK code bases are shown in Table 1.

	Source files	**Lines of code**
MCANO	1799	649336
MONK	878	368709

Table 1: Size of the MONK and MCANO code bases.

MONK, in common with the other MCANO codes, is distributed to users only in the form of a binary executable. Each release of MONK is accompanied by a compatible VRFORT ray-tracing library, supplied as a shared library which can be loaded dynamically by Visual Workshop, the ANSWERS tool for editing inputs, visualizing geometries, launching calculations and displaying results. MONK is commercial software which is licensed to users using a FlexNet Publisher licence. To use MONK within the NURESIM platform users will therefore need a valid licence file. It should be further noted that MONK is subject to export control, and it is therefore necessary to apply for a UK Government export license before it can be supplied to any user outside of the United Kingdom. MONK is developed and tested on both Linux and Windows platforms, but for the purposes of the McSAFE project it is the Linux platform which is of most interest.

2.3. Integration

The NURESIM platform has been developed since 2005 as a reference framework to provide high quality software tools for safety analysis of light water reactors. It is based upon the SALOME

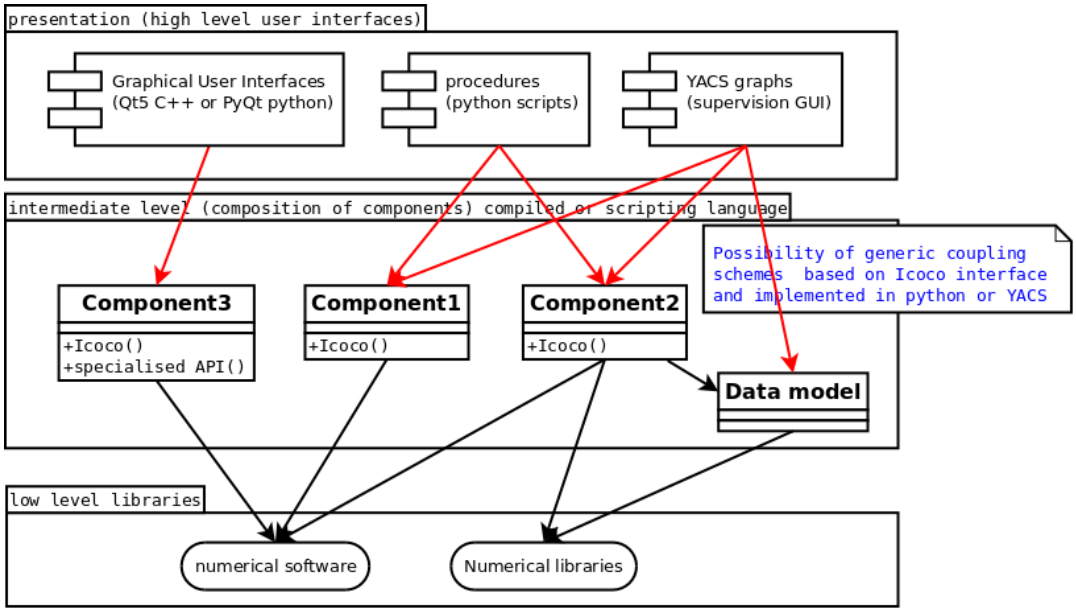

Figure 1: SALOME three-level architecture [9].

platform, which is a generic, open source integration platform for numerical simulation.

The platform is provided with native modules which handle pre-and-post processing and code coupling. An important feature of the platform is that it is possible to integrate numerical solvers in order to couple them to each other and to the native modules. Computational codes are integrated in the platform as components, and can then be coupled together with a *supervision* module. The platform is developed in C++ although all of its interfaces are also exported to Python [9].

SALOME has a modular architecture based on the notion of a *component*, which is an indivisible software unit that provides services accessed through an interface, and can be composed of other components. The platform has a multi-layer architecture, which can be classified into three main levels:

1. This is the lowest level and contains simulation codes that can be written in any language as long as they are compatible with C++ (i.e. they can expose an interface which can be called from C++). In order to facilitate the integration, the software is provided as a shared library, which contains all the functions of the software. These functions will be then called by the intermediate layer for the implementation of the component interface.
2. The intermediate level is a C++ class, which declares and implements the interface of the component. The implementation will call the lower level. The purpose of the component is to drive the underlying code and to exchange numerical fields and meshes with other codes.
3. The highest level is the presentation tier and is dedicated to the users. The interfaces provided at this level can be graphical (GUI) or textual (TUI). GUIs should preferably be implemented in a compiled language (mainly for sake of efficiency).

The generic platform interface is called ICoCo (Interface for Code Coupling) and it is based on the notion of a *Problem*. A Problem is seen as an engine which computes a time-dependent solution, taking as its inputs time-dependent data.

ICoCo is written in C++ and defines an abstract interface that will be implemented by the simulation components. It defines methods that allow initialization and termination of the code, time advance and sub-iterations, saving and restoring and field exchange [9].

Since the MONK executable is built as a monolithic application is has been necessary to refactor the code into a dynamic shared library in order to integrate it into NURESIM. This refactoring mainly involved recasting the MONK main program into a series of Fortran functions and subroutines with ISO C binding so that they can be called from C++. These Fortran functions and subroutines implement the required ICoCo interface and enable MONK calculations to be run by calling these interface routines from C++ instead of running the MONK main program. Since most of the core MONK functionality is provided by Fortran modules which did not need any modification, the refactoring required to integrate MONK into NURESIM was relatively straightforward. In order to couple MONK with other codes (primarily the SUBCHANFLOW subchannel thermal hydraulics codes) it was also necessary to provide methods to exchange data in the specific MED format used by NURESIM/SALOME.

2.3.1. Version control

Prior to commencing any development work a new SVN repository was created specifically for the McSAFE project. This is located on the same secure network as the MCANO repository in which the MONK code is developed. From this network it is not possible to access external SVN or Git repositories, so it is not possible to link directly to the McSAFE GitHub.

Rather than copying the MONK source code into the McSAFE repository directly, the svn:externals mechanism is used to map the trunk of the MCANO repository to a directory in the McSAFE repository. This is done for two reasons:

- to ensure the shared library is always based on the current version of the MONK source, which is subject to continuous integration testing; and
- to make a clear distinction between standard, verified and validated MONK code and code which needs to be modified for the McSAFE project.

As the underlying MONK functionality is being added to NURESIM in the form of a dynamic shared library it is straightforward to provide the latest version to end users at any point without needing to to rebuild the NURESIM platform.

2.3.2. ICoCo methods

A requirement of the integration approach is that the numerical codes should implement the ICoCo integrated code coupling methods as far as possible, and as far as is necessary. The significant methods for MONK are briefly described here. As MONK is a Fortran code most of these methods

have been implemented in Fortran with ISO standard C-bindings, and with explicit C binding names rather than relying on *name mangling* [†].

`setDataFile()`: Gives the name of a data file to the code. This optional method is used to pass a *datsets* file name to the MONK module. The datsets file is the standard mechanism used by MONK to identify input, output and data files. If omitted a default datsets file is expected in the current directory.

`setMPIComm()`: Gives an MPI communicator to the code, for its internal use. This method is used to tell the MONK library which MPI communicator to use. This enables MONK to use the supplied communicator rather than MPI_COMM_WORLD.

`initialize()`: Initializes the code using the data provided by calls to `setDataFile()` and `setMPIComm()`. Initialization of MONK includes setting up initial flags and data, starting the main timer, reading the datsets file and opening the input, output, error and stop files.

`terminate()`: Terminates the computation, frees the memory and saves whatever needs to be saved. Cleans up and deallocates arrays, so that `initialize()` can be called again for a subsequent time step.

`solveTimeStep()`: Performs the computation on the current interval, using input fields. MONK features an internal looping mechanism enabling a range of calculations to be performed in a single execution from a single input file with one or more varying parameters. The `solveTimeStep()` method controls the initialization of the data which need to be reinitialized in each loop, reads the input file, distributes the data to MONK modules, runs the calculation and produces the results. It returns false if the calculation fails. After this call (if successful) the solution for the current time step is accessible in the output file.

`validateTimeStep()`: Validates the computation performed by `solveTimeStep()`.

`getInputFieldsNames()`: Returns a list of strings identifying input fields.

`getOutputFieldsNames()`: Returns a list of strings identifying output fields.

`setInputField()`: Provides a named input field to the code. The method takes as arguments: the field name, the MEDCoupling field and its units. Internal methods provide the necessary formulas to convert between different units.

`getOutputField()`: Gets a named output field from the code. This method returns the MED-Coupling output field and takes as arguments the field name and its unit. Internal methods provide the necessary formulas to convert between different units.

`setRequestedLoop()`: This optional method is used to select a specific loop in a MONK looping calculation. Loop selection is achieved using a command line option in the standard version of

[†]Name mangling is the encoding of function and variable names into unique names so that linkers can separate common names in the language.

MONK so this method is required to replicate this functionality in the NURESIM framework.

`suppressTraces()`: This optional method is used to suppress the adding of burnup traces to materials in a burnup calculation. In the current MONK version, the power distribution can only be calculated in the right form within a burn-up calculation. The simulation can be accelerated using this method if the burnup products are not actually required. A future development to MONK will provide the ability to calculate the power distribution without doing a depletion calculation.

`setDebugOption()`: Prints additional information about the nuclear mesh and the fields generated by MONK.

`getNuclearMesh()`: Generates the nuclear mesh based on information coming from the MONK input. It takes as an argument the nuclear mesh unit. Internal methods provide the necessary formulas to convert between different units.

`getMonkMeshgrid()`: Retrieves MONK's meshgrid information. This information is based on MONK's thermal hydraulics input meshgrid definition and is used to generate the nuclear mesh.

`checkCellVolumes()`: This methods is used to print the volumes of each cell contained in the nuclear mesh. This method is triggered by `setDebugOption()`.

`checkFieldValues()`: This method prints the field values belonging to the nuclear mesh. This method is triggered by `setDebugOption()`.

`checkCellBarycenters()`: This method is used to print the barycenters of each cell contained in the nuclear mesh. This method is triggered by the `setDebugOption()`.

`getMonkPowerDistr()`: This method returns the power distribution calculated by MONK. Currently the calculation of the power distribution can only by triggered by a burnup calculation.

`createCoordinates()`: This is an auxiliary method that sets the coordinates to create the nuclear meshgrid.

2.3.3. MONK Shared Library

In order to integrate MONK into the NURESIM platform it has been necessary to convert it from a standalone executable to a shared library, as described in Section 2.3. Building a static library is part of the standard MONK build since this is used by the MCANO Fortran unit test framework. Initial testing of the ICoCo methods was therefore carried out using this static library. However it is preferable to have a dynamically-loaded library which can be loaded and unloaded as required. This also allows the library to be updated without needing to relink with the platform. Therefore the MONK library has also been built as a 64-bit dynamic shared library.

2.3.4. Technical Challenges

It might be expected that refactoring a mature Fortran code like MONK into a dynamic shared library with C++ interfaces might be a significant challenge. However, this aspect of the integration

of MONK into NURESIM was relatively straightforward. Here the technical issues mostly related to ensuring that memory was deallocated when no longer needed. In a standard MONK execution this is not an issue since the memory is automatically released when the application terminates. But in the NURESIM platform the MONK library is loaded dynamically, requiring the memory to explicitly be released when no longer required. These issues were solved relatively easily.

A pre-requisite for coupling MONK into the NURESIM platform is downloading and building the platform itself, and this is not straightforward. NURESIM requires a large number of external dependencies and makes assumptions about the operating system and the availability of these dependencies either via a network to a Git repository or on local disks. NURESIM has the "SAT" tool that attempts to provide configuration control on these external dependencies as NURESIM is built. However, a significant number of changes to the configuration were required for this to be successful. Given the freedom to use the recommended operating system and access external repositories many of the difficulties may have been overcome, but this kind of unconstrained IT freedom is rare in the nuclear industry for reasons related to information security.

The specification of the McSAFE project requires code developers to integrate their numerical codes in the NURESIM platform using the SALOME three-level architecture described in Section 2.2. The use of the MEDCoupling library to exchange fields between codes and, to some extent, the use of a standard ICoCo interface, is valuable and relatively straightforward. However, the production of intermediate level components and high level supervisors within the NURESIM/SALOME platform adds considerable complexity to the task of coupling numerical codes.

2.4. Testing

2.4.1. Fortran Unit Testing

The newly-implemented ICoCo methods have been tested using the ANSWERS Fortran unit test framework. The unit test framework is linked with the static MONK shared library. These unit tests are designed to check that the ICoCo methods correctly interface with the MONK library, and that the data are passed correctly.

2.4.2. C++ Test Program

The ICoCo interface routines have also been tested using a C++ test program which calls the underlying Fortran routines. This test program dynamically loads the MONK shared library and runs complete MONK calculations by calling the ICoCo methods which have been exposed to C++. The results have been compared with results from the released version of MONK10B to verify that calculations run with the shared library agree with those run with the standard version of MONK.

This C++ test program serves a further purpose in that it can mimic a standalone MONK execution by processing command-line parameters in the same way as the standalone MONK executable. This means that we can submit this executable to the ANSWERS Automated Testing Tool (see Section 2.4.3) as we do when testing standalone MONK.

2.4.3. ANSWERS Automated Testing Tool

The ANSWERS Automated Testing Tool (ATT) is an in-house product used for continuous integration testing of all ANSWERS codes. The ATT runs a requested set of test cases using the executable being tested, compares the results with verified reference results using a set of defined rules and automatically generates a test report. This integrates with a system known as buildbot, which also integrates with the source code Subversion repository. Whenever a code change is committed to the repository buildbot automatically builds all affected codes on all target platforms then runs a subset of the code tests using the ATT and generates a set of test reports. For MONK this currently runs 211 unit tests and 172 ATT integration tests after each code commit. Each night a much larger set of tests is run (currently over 3600 tests for MONK).

As described in Section 2.4.2, the C++ test program can be submitted to the ATT to test the McSAFE MONK shared library against the standard MONK test sets. These include tests for a wide range of MONK functionality, including functional testing of the collision processing for every nuclide in each of the following nuclear data libraries: CENDL-3.1; ENDF-B/VII.0 and ENDF-B/VII.1; JEF-2.2; and JEFF-3.1, JEFF-3.1.1, JEFF-3.1.2 and JEFF-3.2.

3. VERIFICATION

The verification of the coupled code system is carried out performing a comparison between MONK and SUBCHANFLOW coupled via NURESIM, and the WIMS/ARTHUR suite. WIMS is a modular, deterministic reactor physics code suite and ARTHUR is a new subchannel thermal hydraulics module developed specifically for WIMS [10]. Therefore the WIMS/ARTHUR suite provides a completely independent calculation against which the MONK/SUBCHANFLOW calculations can be compared.

3.1. Benchmark Specification

A set of benchmarks tests has been proposed for verification of the MONK thermal hydraulics coupling mechanism. Here we discuss the simplest: a single PWR pincell. The parameters defining this pincell are shown in Table 2.

Parameter	Value
Pellet radius	0.4095 cm
Clad radius	0.4750 cm
Rod pitch	1.2600 cm
Active length	366.00 cm
Power	0.0670 MW

Table 2: PWR pincell benchmark parameters

The fuel pin is modelled in MONK with reflective boundary conditions in radial directions and vacuum ones in the axial directions, and it is subdivided into 10 axial divisions to represent material temperature profiles and the coolant density profile.

3.2. Comparison with WIMS

The PWR pincell case has been modelled using WIMS coupled with its own sub-channel module ARTHUR [11] and the results are shown in Figure 2.

Both coupled schemes use an iterative approach in which the neutronic and the thermal-hydraulic calculations are iterated until the designated maximum error is reached on a specific set of parameters. For the NURESIM coupling the exit condition is the error on the multiplication factor while for WIMS/ARTHUR it is on the fuel temperature. Furthermore, for both coupled schemes a relaxation factor of 0.6 has been chosen and it has been applied to all the temperature and density fields.

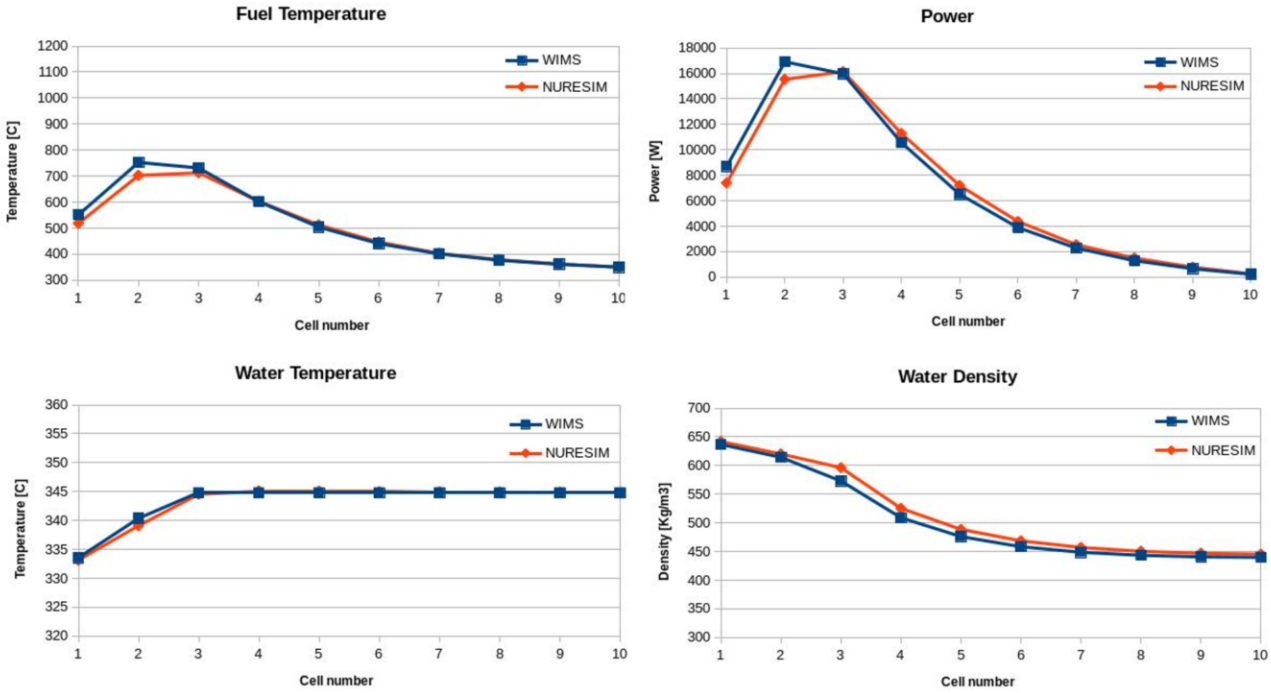

Figure 2: PWR pincell benchmark results based on MONK-SCF coupling within the NURESIM platform, compared with WIMS results.

With respect to the neutronic simulation in WIMS, the neutron cross sections have been generated by the collision probability module FLURIG and, after being homogenized, they have been passed to the newly-developed module MERLIN. This module, based on the SP_3 approach [12], provides the power distribution needed by the sub-channel code to calculate the temperature and density distributions.

Where possible, ARTHUR calculation schemes and correlations were aligned to SUBCHAN-

FLOW ones. However, the two codes do not share an equivalent two-phase friction correlation so that the EPRI one has been chosen for ARTHUR and the Armand one for SUBCHANFLOW [13]. Other phenomena, like subcooled boiling, have been not modelled at all for the same reasons.

Additionally, ARTHUR has material laws that differ from the ones that can be found in SUBCHANFLOW, therefore, appropriate fuel and clad temperature-independent parameters were chosen in both codes in order to carry out the comparison.

Taking into accounts differences concerning correlations, physics models and convergence criteria, it can be said that WIMS-ARTHUR results agree well with the NURESIM ones and this, in turn, demonstrates the correctness of the coupling implementation and of the physics models involved.

4. CONCLUSIONS

MONK has been successfully refactored in the form of a dynamic shared library, using ISO standard C bindings to allow the functionality to be accessed from C++, and tested with a combination of unit testing and automated regression testing, giving confidence that refactoring MONK as a shared library does not introduce any systematic errors. The ICoCo methods necessary for coupling MONK to SUBCHANFLOW have also been implemented and tested, and a C++ class has been implemented to provide the MONK component within the NURESIM platform.

While the integration of MONK into NURESIM has demonstrated that a mature, industry standard, commercial computational tool can be integrated in the platform, further development of the platform could significantly improve its utility as an industrial tool.

MONK has also been coupled successfully to SUBCHANFLOW within the NURESIM platform and the results for the benchmark tests agree well with the WIMS reference calculations. It has therefore been demonstrated that MONK has been successfully integrated into the NURESIM platform and successfully coupled to SUBCHANFLOW within that platform.

5. ACKNOWLEDGEMENTS

This work is being carried out as part of the McSAFE project which is funded by the European Union Horizon 2020 Research and Innovation Framework programme under grant No. 755097 from September 2017 to August 2020.

REFERENCES

[1] C. Chauliac, J.-M. Aragones, D. Bestion, D. G. Cacuci, N. Crouzet, F.-P. Weiss, and M. A. Zimmermann. "NURESIM - A European simulation platform for nuclear reactor safety: Multi-scale and multi-physics calculations, sensitivity and uncertainty analysis." *Nuclear Engineering and Design*, **volume 241**, pp. 3416–3426 (2005).

[2] http://www.salome-platform.org.

[3] S. D. Richards, C. M. J. Baker, P. Cowan, N. Davies, and G. P. Dobson. "MONK and MCBEND: Current Status and Recent Developments." *Annals of Nuclear Energy*, **volume 82**, pp. 63–73 (2015).

[4] S. D. Richards, M. Shepherd, A. Bird, D. Long, C. Murphy, and T. Fry. "Recent Developments to MONK for Criticality Safety and Burnup Credit Applications." In *Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*. Jeju, South Korea (2017).

[5] S. D. Richards, G. Dobson, D. Hanlon, R. Perry, F. Tantillo, and T. Ware. "MONK11A: Status and Plans for the MONK Monte Carlo Code for Criticality Safety and Reactor Physics Analysis." In *Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*. Portland, Oregon, USA (2019).

[6] R. J. Brissenden and A. R. Garlick. "Biases in the Estimation of k_{eff} and its Error by Monte Carlo Methods." *Annals of Nuclear Energy*, **volume 13**, pp. 63–83 (1986).

[7] E. Woodcock, T. Murphy, P. Hemmings, and S. Longworth. "Techniques used in the GEM Code for Monte Carlo Neutronics Calculation." In *Proceedings of Conference on the Applications of Computing Methods to Reactor Problems*, ANL-7050, pp. 557–579.

[8] http://www.answerssoftwareservice.com.

[9] N. Crouzet. "Specification of the MC code integration and coupling interface." McSAFE EU project.

[10] B. A. Lindley, J. G. Hosking, P. J. Smith, D. J. Powney, B. S. Tollit, T. D. Newton, R. Perry, T. C. Ware, and P. N. Smith. "Current status of the reactor physics code WIMS and recent developments." In *Proceedings of Conference on the Applications of Computing Methods to Reactor Problems*, Annals of Nuclear Energy, pp. 148–157.

[11] B. Tollit, A. Charles, A. Cox, K. Kohli, B. Lindley, G. Hosking, P. Smith, M. Dillistone, and P. Smith. "DEVELOPMENT OF A SUBCHANNEL MODEL WITHIN THE ANSWERS SOFTWARE SERVICE WIMS REACTOR PHYSICS CODE." In *Proceedings of Conference on the Applications of Computing Methods to Reactor Problems*, PHYSOR 2018.

[12] W. F. M. E. E. Lewis. "Computational methods of neutron transport." (1984).

[13] U. Imke. "Input Instructions for SUBCHANFLOW 3.0." (2015).