# Design and implementation of a radiation transport code input generation interface

Chance B. Loveday (Maryville College, Maryville, TN 37804)
Douglas E. Peplow (Oak Ridge National Laboratory, Oak Ridge, TN 37830)
Robert A. Lefebvre (Oak Ridge National Laboratory, Oak Ridge, TN 37830)

July 2023

## Abstract

Accurate modeling of radiation transport is essential for ensuring the safety and reliability of nuclear reactors. To achieve the necessary level of precision and prevent unforeseen incidents during experiments, modeling software such as MAVRIC or ORIGEN under the SCALE code system requires input that describes the system in which to simulate the behavior of radiation transport. The Workbench Analysis Sequence Processor (WASP) project provides data structures and algorithms, written in C++, that can efficiently enable data access and processing. Our project utilizes the WASP toolset to produce problem-specific material inputs for radiation transport codes. Each database is JSON-formatted and contains an array of material definitions that specify the elemental or isotopic ratios for each material. JSON formatting permits the utilization of a universal format and the data structures built into WASP in structuring the material composition data. The components from WASP allow classes to access the databases and to construct materials and components into objects based on the properties described in the material composition database files. The implementation includes methods for building, checking, and translating this information into an acceptable format for the modeling software. Furthermore, the code structure can be integrated into a graphical user interface, offering enhanced accessibility and ease of

use. The material composition interface can potentially reduce mistakes caused by manual entries and increase simulation accuracy by applying more realistic problem-specific material definitions. This approach also supports composition conversions into various formats for MAVRIC, Keno, ORIGEN, MCNP, and a generic nuclear code input. As a result, access to atomic information and software compatibility improves. Thus, our project's products provide an accessible, accurate, and error-free approach to generating material descriptions for radiation transport simulations.

## I Introduction

Radiation transport and modeling of nuclear reactors are primarily performed and understood through modeling software. These tools employ continuous testing and calculations to identify and analyze the intricate reactions involved, ensuring precision and a thorough understanding before physical experiments are conducted within nuclear facilities. One crucial aspect of modeling software is input generation, which can become tedious and time-consuming, especially if multiple versions of each input are being prepared. Systems such as SCALE Monaco with Automated Variance Reduction using Importance Calculations (MAVRIC), Keno, and ORIGEN[1] are commonly used for radiation transport and shielding experiments. Manual inputs require users to search for material information, likely from several isolated sources. Therefore, a tool capable of generating job-specific radiation transport code inputs would save researchers time and prevent material input errors. Previous students attempted to develop a stand-alone graphical user interface (GUI) to streamline handling such inputs for radiation transport and shielding calculations. Although a GUI using Java successfully launched, the need for accessibility to atomic information has grown as SCALE and Monte Carlo N-Particle Transport Code (MCNP)[2] have released newer versions.

Given these considerations, this project was revisited to provide a convenient, more user-friendly tool for generating radiation transport code material inputs. To achieve this goal, the input generation tool would need to interpret the material composition data from various JavaScript Object Notation (JSON) files from different references. Each file contains information regarding the composition and radioactive materials. The Workbench Analysis Sequence Processor (WASP)[3] project provides data structures and algorithms that efficiently enable the data access and processing required to parse and interpret the databases. The new tool's functionality will involve building and preparing each piece of information into a defined object. The interface will operate off a series of methods that build, display, and prepare the data for transformation into formatted input. To ensure the quality and accuracy of the interface, the interface will need to undergo extensive testing before it can become available as a GUI. Through the availability of a more accessible tool for a wide range of users, this project strives to tackle the challenges associated with input generation in radiation transport simulations to continue advancements in this field. Materials information from several references, as shown in Table $1^{4-18}$, were converted to the JSON format to ensure the universal storage of information.

Table 1: Materials Database from Various Sources

| Description | Materials Listed<br>Composition and Density | <br>Composition Only | <br>Density Only |
|---|---|---|---|
| PNNL Compendium [4] | 411 | | |
| Shultis & Faw text book [5] | 7 | | |
| Criticality Handbook, ARH-600 [6] | 61 | | |
| Pu mixes, DOE-STD-3013-2018 [7] | | 7 | |
| ICRP 89 Human biology [8] | 41 | | |
| ICRU 44 tissues and substitutes [9] | 93 | 3 | |
| ICRU 46 Human tissues [10] | 106 | | |
| Marks' Standard Handbook [11] | | | 217 |
| Crit. Calcs. MCNP5 Primer [12] | 34 | | |
| NIST Database 124 [13] | 279 | | |
| NIST Database 126 [14] | 48 | | |
| Reactor Hndbk, Chap 19-48 [15] | 246 | | 2 |
| Reactor Hndbk, Chap 51 [15] | 381 | | 135 |
| SCALE manual Tbl 7.2.5 [16] | 20 | | |
| SCALE manual Tbl 7.2.4 [16] | 46 | | |
| CRC Inorganic Compounds [17] | 2151 | 1069 | |
| CRC Organic Compounds [18] | 5566 | 5301 | |

# II Methods

## A. Material composition files and building objects from databases

WASP recognizes seven distinct data types in storing materials information but places a particular emphasis on data objects and data arrays. These data types are described in the WASP JSON Parser[3]. Data objects are similar data structures to dictionaries or maps, essentially mapping an object to its identifier (i.e. “Name” or “Density”); on the other hand, data arrays store an array of data objects. Material composition JSON files can contain four types of objects that can be categorized into classes: databases, materials, components, and contacts. ”Component” and ”Contact” are members of ”Material,” and the database class holds a vector of material objects. The classes are established with variable initialization, constructors, set methods, get methods, and additional methods as needed. Databases are intended to hold standard criteria, including a database name, reference, material names, material densities, material types, and material components. Databases such as PNNL[4] provide additional properties but are optional fields and may not exist in every database. Most materials are read straightforwardly, whereas others require specific conditionals. For example, a material could contain a reference stored as a string or an array of references.

The class structure defines a token as a member of one of the defined classes that must be assigned to a class member through a build. The database class stores a series of build functions that read the data objects and assign data values to a data member based on the token and data type. The head build function takes a path for the masses database and an error stream; the masses database builds via a series of functions autonomous to the Database class, and the WASP JSON Object Parser is called. Each data object or data array undergoes a corresponding build, so the build function relies on the database build. Similarly, a build relies on the construction of subsequent components and continues until it meets crucial criteria or finishes successfully. The build functions utilize a concept known as ”Design by Contract,” where specific interface conditions must be met. In this case, the build functions must all return true to verify that the file is JSON-formatted and fulfills the requirements of the material database. WASP components provide the proper utilities to parse and test the files. Examples of some material definitions (serpentine from Shultis&Faw1996[5], zeolite from PNNL, and water from ARH-600[6]) are shown as follows.

```
{ "Name":  "Serpentine",
  "Comment":  "d Serpentine (3MgO?2SiO2?2H2O) as aggregate; a
concrete usable at high temperatures with minimal water loss.",
  "Density":  2.1,
  "Type":  "Weight Fractions",
  "Contains":  [
     {"Element":  "H", "Fraction":  0.015909 },
     {"Element":  "O", "Fraction":  0.511818 },
     {"Element":  "Si", "Fraction":  0.209091 },
     {"Element":  "Ca", "Fraction":  0.068182 },
     {"Element":  "C", "Fraction":  9.09E-4 },
     {"Element":  "Na", "Fraction":  0.004091 },
     {"Element":  "Mg", "Fraction":  0.135 },
     {"Element":  "Al", "Fraction":  0.019091 },
     {"Element":  "K", "Fraction":  0.004091 },
     {"Element":  "Fe", "Fraction":  0.030909 },
     {"Element":  "Cr", "Fraction":  9.09E-4 }
  ]
}

{ "MatNum":  401,
  "Name":  "Zeolite (Natrolite)",
  "Aliases":  ["Natrolite"],
  "Abbreviations":  ["369"],
  "Density":  2.25,
  "Type":  "Atom Fractions", "Formula":  "Na2Al2Si3O12H4",
  "Contains":  [
     {"Element":  "H", "Fraction":  "0.17391304347826086" },
     {"Element":  "O", "Fraction":  "0.5217391304347826" },
     {"Element":  "Na", "Fraction":  "0.08695652173913043" },
     {"Element":  "Al", "Fraction":  "0.08695652173913043" },
     {"Element":  "Si", "Fraction":  "0.13043478260869565" }
  ],
  "Comments" :  "CAS No.  1318-95-2",
  "VerifNotes" :  "Confirmed November 12 2019.",
  "References":  [
     "Formula and density from http://www.webmineral.com",
     "Non-clumping cat litter is often made of zeolite, diatomaceous
earth, and/or sepiolite.  The formula is for natrolite (http://www.galleries.
which is one form of the mineral group called zeolite (http://en.wikipedia.or
  ],
  "Source": "BNL",
  "Contact": {
     "Name": "Giuseppe Camarda",
     "Phone": "631-806-4935",
     "Email": "giuseppec@bnl.gov" }
},

{ "Name": "Water",
  "Density": 1.0,
  "Type": "Chemical Formula",
  "Formula": "H2O",
  "Contains": [
     {"Element": "H", "Atoms": 2.0 },
     {"Element": "O", "Atoms": 1.0 }
  ]
},
```

## B. Internal checks

In addition to defining and building objects, the project comes equipped with internal checks to ensure that the compositional data is consistent throughout the file. The method "check" takes a material and calls a method that checks the sum of the fractional components or the atom counts based on the composition type. The former confirms that the fractions add to one with five decimal places of precision. The "check atoms" method dissects the given formula–accessible only if the material was originally stored as atomic fractions or a chemical formula–into atom amounts. Atom counts derived from the formula are concatenated into a string and compared with a string of the atom amounts specified in the components. Formulas with several multipliers and nested parentheses complicate and delay the process of retrieving the correct number of atoms. Although this process may seem arduous and unnecessary, checks and tests provide quality assurance to the users inside the laboratory and external clients. Design by Contract and internal checks expose discrepancies and help reassure data marshals and developers that the software is ready for deployment; one typo has already been corrected in a JSON file as a result of this check.

## C. Composition conversions

The masses class helps to transform and prepare the data into a desired format for the user. Similar to the database class, the ”Masses” class searches for a name, reference, notes, and a list of materials, yet it references different types of files. Each material is linked to an atomic number, a symbol, a list of isotopes, and a mass, if the isotope is naturally occurring. The masses class has a similar series of build functions, so mass JSONs are referenced primarily for their atomic masses and isotopic information. This program intends not only to provide a quicker and more accessible means of generating radiation transport code inputs but also to convert between composition types. The different JSON databases contain composition information in one of three forms: weight fractions, atomic fractions, or atoms per molecule based on a chemical formula. Given a compound’s atom fractions $a_z$ and atomic masses $M_z$ for each element $z$, the weight fractions $w_z$ are found using the equation

$$w_z = \frac{a_z M_z}{\sum_i (a_i M_i)}.$$

Conversely, atom fractions are found by the equation

$$a_z = \frac{\frac{w_z}{M_z}}{\sum_i \frac{w_i}{M_i}}.$$

Given atomic fractions or atoms per molecule, the following formulas convert between the two respectively. Note that $p$ represents atoms per molecule.

$p_z = n(\frac{a_z}{m}) \ni m$ is the minimum at.% and $n$ is the molecular coefficient.

$$a_z = \frac{p_z}{\sum_i p_i}.$$

The coefficient for the chemical formula $n$ is found by dividing the atom amounts in the formula by the empirical atom amounts, building a molecular formula. Weight fractions can also be found by multiplying the element’s atoms per molecule by the given mass and dividing by the total sum of the products.

$$w_z = \frac{p_z M_z}{\sum_i p_i M_i}.$$

However, weight fractions or atom fractions cannot be converted to atoms per molecule without a chemical formula because of the uncertainty of finding a molecular formula without a formula or molar mass. Finally, a copy of ”Contains” called ”native” is stored in memory when the database is built, so the compositions can be reset to their native state.

Conversions among these three forms provide a more understandable, user-friendly input, open the opportunity to include or exclude isotopic information, and ensure consistency among the generated inputs. The conversion between elemental and isotopic formats occurs when the elements are atomic fractions. Either the atomic fractions are multiplied by each isotopic natural abundance and added to the “Component” vector “Contains,” or the isotopic atom fractions are summed into one fraction per element and removed from “Contains.” Note that the elements are split into isotopes only if the element was not originally assigned a mass number. A crucial distinction exists between elemental and isotopic information. If a mass number is not assigned for a given element, the isotopic breakdown is assumed to consist of naturally occurring abundances. Otherwise, the isotopes specified in the database are retained. Photon scattering calculations require only the elemental information of a material, whereas neutron scattering calculations heavily depend on the isotopes present in the material.

Four distinct input formats are supported by this project: MAVRIC/Keno, ORIGEN, MCNP, and Generic. As an example of the variations of the input formats, consider the compound acetone. MAVRIC/Keno inputs, as shown in the following lines, typically consist of one line more than the number of elements present. The first line starts with the type concatenated with up to 12 characters from the name, followed by a labeling number, density, and the number of elements. Each subsequent line contains the elements ZAID (1000*atomic number + mass number), and the composition amount; the last line finishes with “end.”

```
′ PNNL-15870Rev2
′ Acetone, C3H6O, 0.7899 g/cm³
′ The above density is estimated to be accurate to 4 significant
digits.  Uncertainties are not addressed.
′  The following data was calculated from the input formula.
   wtptAcetone 1  0.7899  3
       1000   10.4131
       6000   62.0392
       8000   27.5476  end
```

ORIGEN takes inputs in blocks, with isotopes listed inside curly braces on one line followed by the units.

```
mat {
iso= [ H=0.104131
       C=0.620392
       O=0.275476 ]
units=grams
}
```

MCNP takes an ID number and then the ZAID and composition amount for each material. In MCNP, weight fractions are expressed as negative values and atom fractions as positive values to clearly distinguish between the two.

```
c  PNNL-15870Rev2
c  Acetone, 0.7899 g/cm3
c  The above density is estimated to be accurate to four significant
digits.  Uncertainties are not addressed.
c  The following data was calculated from the input formula.
      m1 1000  -0.104131
         6000  -0.620392
         8000  -0.275476
```

The generic format is designed for similar codes within SCALE to adhere to a standard; each line of the input takes an atomic number, a symbol, and a mass number.

```
#  PNNL-15870Rev2
#  Acetone, 0.7899 g/cm3
      1   H   0    0.104131
      6   C   0    0.620392
      8   O   0    0.275476
c Generic Weight Fractions
```

# III Results

## A. GUI stand-in

Due to time constraints, a stand-in for a GUI was developed in C++ to simulate the program's capabilities. The interface is a class called "Material Composition Lib," which requires a parent directory containing a masses file and at least one material composition file. After building the databases and masses, the user is given a list of the available materials and asked to pick one based on the corresponding index. This option emulates a scrollable list of materials provided by a GUI. The user picks from a printed list of available commands referenced by an integer with -1 corresponding to "quit." Seven basic commands are executed through a for loop and switch statements. The first two commands call the display methods from the masses database and the specified material, respectively. The search feature takes in a string from the user and returns all materials that contain that string, simulating a search bar. The user is then prompted to change materials; there is also an independent option to change materials. The interface also retains an advanced search feature that searches for a material based on various parameters, including index, name, and density. The user can check the components by calling the Material class check functions. Finally, there is an option to create a transport code input, which takes parameters for the desired system format, composition type, and calculation type. A GUI would ideally have radio buttons for the various selection options.

## B. Future work

The material composition database project could continue in the hands of future interns or ORNL staff members to further implement the program into a GUI. Previous work on this project saw the deployment of a GUI written in Java to calculate inputs based on material compositions found in XML database files. The 2015 Java GUI was functional but was developed as a tool entirely independent of SCALE or MCNP. Java is also an interpreted language, so compiled languages like C++ would be favorable for large-scale conversions or scripts. By nature, C++ scripts would run notably faster than scripts written in Java or Python. Additionally, the XML format is more difficult for developers and users alike to decipher, causing JSON files to be used instead. Git version control was instrumental in developing and sharing updates with the research group. Semantic versioning and branch workflow track updates among project members via GitLab. All commits must be approved and must not fail any scheduled pipelines before versions merge with the master version. Ultimately, the input generation tool will be handed off to the Nuclear Code Integration Group for the implementation of the interface into the relevant modeling software. SCALE, ORIGEN, and MCNP users would then have access to a built-in material composition input tool that would avoid the need to download and use a stand-alone tool.

## IV Conclusion

The development of a user-friendly input generation tool represents a significant step forward in streamlining the radiation transport simulation process and facilitating the realization of accurate results. An efficient, intuitive GUI would alleviate the drawbacks of manual input preparation. The reduction in steps offered by a GUI would save time and energy. Input generation as designed during this project would make processes smoother and credible information obtainable from one location. The future progress of this project will see a visual representation of this interface built into popular systems, such as SCALE software or MCNP. Accordingly, errors recorded in databases should be easier to catch by the time the code is integrated. Users of MAVRIC, Keno, ORIGEN, MCNP, and similar programs would find this input generation more accessible and time-efficient compared with past efforts.

## Acknowledgements

This research was supported by the U.S. Department of Energy, Office of Science, and Oak Ridge National Laboratory, Nuclear Energy and Fuel Cycle Division under the Science Undergraduate Laboratory Internship program.

## References

1. William A. Wieselquist and Robert A. Lefebvre, editors, *SCALE 6.3.0 User Manual*, ORNL/TM-SCALE-6.3.0, Oak Ridge National Laboratory, Oak Ridge, Tennessee, December 2021. doi.org/10.2172/1909149

2. Jerawan Armstrong, *et al., MCNP User's Manual – Code Version 6.2*, LA-UR-17-29981, Edited by Christopher J. Werner, Los Alamos National Laboratory, Los

Alamos, NM, October 27, 2017. https://mcnp.lanl.gov/pdf_files/TechReport_2017_LANL_LA-UR-17-29981_WernerArmstrongEtAl.pdf

3. Lefebvre, Robert A., Langley, Brandon R., Lefebvre, Jordan P., & USDOE. (2017, November 16). Workbench Analysis Sequence Processor [Computer software]. https://www.osti.gov//servlets/purl/1419488. https://doi.org/10.11578/dc.20201125.2

4. *Compendium of Material Composition Data for Radiation Transport Modeling*, 200-DMAMC-128170, PNNL-15870, Rev. 2, Pacific Northwest National Laboratory, Richland, Washington, April 2021. https://doi.org/10.2172/1782721 (https://compendium.cwmd.pnnl.gov/)

5. R. D. Carter, G. R. Kiel, K. R. Ridgway, *Criticality Handbook*, Volume I, ARH-600, Atlantic Richfield Hanford Company, Richland, Washington, June 30, 1960.

6. DOE-STD-3013-2018, DOE Standard: *Stabilization, Packaging, and Storage of Plutonium-Bearing Materials*, U.S. Department of Energy, Washington, D.C., November 2018.

7. J. Valentin, Editor, *Basic Anatomical and Physiological Data for Use in Radiological Protection: Reference Values*, ICRP Publication 89, The International Commission on Radiological Protection, Published by Elsevier Science Ltd., 2003.

8. *Tissue Substitutes in Radiation Dosimetry and Measurement*, ICRU Report 44, International Commission on Radiation Units & Measurements, January 1989.

9. *Photon, Electron, Proton and Neuron Interaction Data for Body Tissues*, ICRU Report 46, International Commission on Radiation Units & Measurements, February 1992.

10. Eugene A. Avallone, Theodore Baumeister III, and Ali M. Sadegh, editors, *Marks' Standard Handbook for Mechanical Engineers*, Eleventh Edition, 2007.

11. Roger Brewer, *Criticality Calculations with MCNP5: A Primer*, LA-UR-09-00380, Los Alamos National Laboratory, Los Alamos, NM, January 2009.

12. NIST Standard Reference Database 124, Stopping-Power & Range Tables for

Electrons, Protons, and Helium Ions, update July 2017. https://physics.nist.gov/cgi-bin/Star/compos.pl

13. NIST Standard Reference Database 126, X-Ray Mass Attenuation Coefficients, updated July 2004. www.nist.gov/pml/x-ray-mass-attenuation-coefficients

14. *Reactor Handbook*, Second Edition - in Four Volumes, Volume 1: Materials, edited by C. R. Tipton, Jr. Interscience Publishers, Inc., New York, NY, Library of Congress Catalog Card Number 60-11027, 1960.

15. B. T. Rearden and M. A. Jessee, Eds., *SCALE Code System*, ORNL/TM-2005/39, Version 6.2.3, Oak Ridge National Laboratory, Oak Ridge, Tennessee (2018). Available from Radiation Safety Information Computational Center as CCC-834.

16. J. Kenneth Shultis and Richard E. Faw, *Radiation Shielding*, Prentice-Hall, Upper Saddle River, NJ, 1996.

17. 'Physical Constants of Inorganic Compounds,' in *CRC Handbook of Chemistry and Physics*, 100th Edition (Internet Version 2019), John R. Rumble, ed., CRC Press/Taylor & Francis, Boca Raton, FL. [Accessed October 2023.]

18. 'Physical Constants of Organic Compounds,' in *CRC Handbook of Chemistry and Physics*, 100th Edition (Internet Version 2019), John R. Rumble, ed., CRC Press/Taylor & Francis, Boca Raton, FL. [Accessed October 2023.]

19. Coursey, J. S., Schwab, D. J., Tsai, J. J., and Dragoset, R. A. (2015), Atomic Weights and Isotopic Compositions (version 4.1). [Online] Available: http://physics.nist.gov/Comp [2019 October]. National Institute of Standards and Technology, Gaithersburg, MD.