IRSN
INSTITUT
DE RADIOPROTECTION
ET DE SÛRETÉ NUCLÉAIRE

Faire avancer la sûreté nucléaire

Designing an API for reading/writing nuclear data

Wim Haeck, wim.haeck@irsn.fr

Outline

- Introduction
  - Why an API?
  - Our point of view
- Designing an object oriented API
  - Past experience
  - Design choices
  - Example
- Organising the work
- Conclusions

An API because …

- Because we don’t want to reinvent the wheel
- Because we need to be able to construct the new file
- Because we need to be able to use the new file
  - First for testing and visualization
  - Then for general use in our applications

Our point of view

- IRSN is an end user
  - We use basic evaluations to generate our own libraries
  - We use data libraries from the distribution of a calculation code
- Two large C++ projects with direct nuclear data needs
  - VESTA: depletion calculations
  - GAIA: nuclear data processing and formatting
    - Going into the conceptual design phase
    - PhD on resonance reconstruction and Doppler broadening
- Our main interest is the object oriented API in C++ and Java
  - Implementation in both languages can share a common design

Past experience: an ENDF parser

- We use our own ENDF parser in our current software
  - Basic building blocks: ENDFINT, ENDFDOUBLE, ENDFTAB1, ENDFLIST, etc.
  - Construct higher level classes: ENDFMATMF3MT, ENDFMATMF3, ENDFMAT
- Learn from past experience
  - Accessing data is inherently linked to the ENDF data format
    - Not compatible with a new data structure
    - File format and data representation should not be associated
  - Primitive data types are used for input and output
    - Makes it difficult for new people to use the parser
  - Difficult to go back on a design choice
    - Programming is 95% inspiration and 5% transpiration
  - Extremely work intensive
    - Documentation and testing take as much time as implementation

Design choices to be made

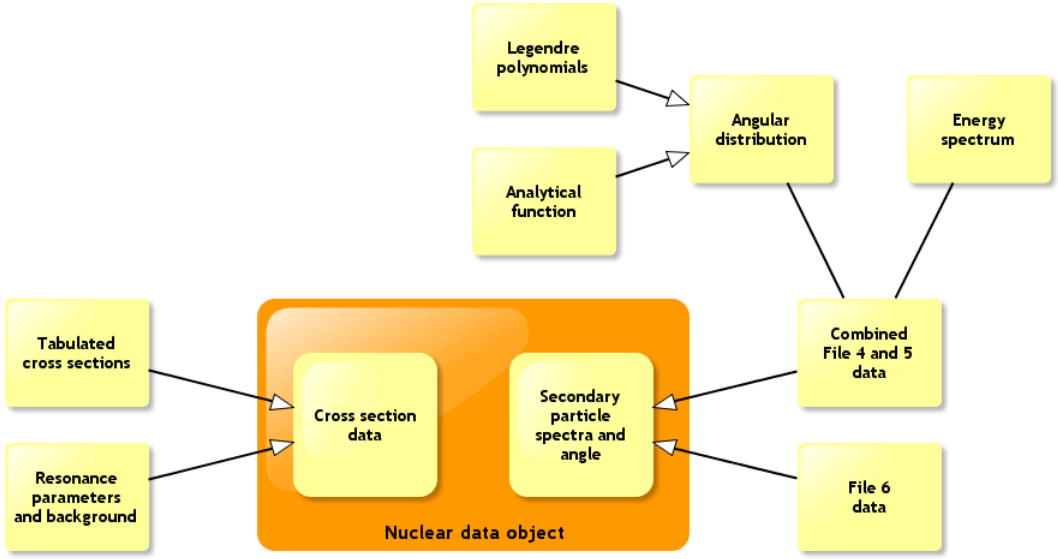
- Low level or high level?
  - Use only low level or primitive data types as input and output
    - For example: use “double” and not the concept of an energy value
    - Can be integrated easily into other software (both existing and new)
  - Use high level concepts and abstraction
    - Includes more of the physics behind the data
    - Independent of the physical file format
    - Goes beyond the scope of a simple API but can be more robust
- Use interfaces?
  - Multiple representation types for the same data
    - For example: secondary particle energy and angular distributions
  - An abstract interface allows for multiple implementations
    - Use a link to physical files or store the data in memory

Design choices to be made

- Where does the API end and where does processing begin?
  - For example: linearisation of data (e.g. for plotting)
    - Which linearisation scheme?
    - What if a user wants to use another scheme?
  - If we add simple operations like this why not add more complex ones?
    - Doppler broadening
    - Multi-group treatment

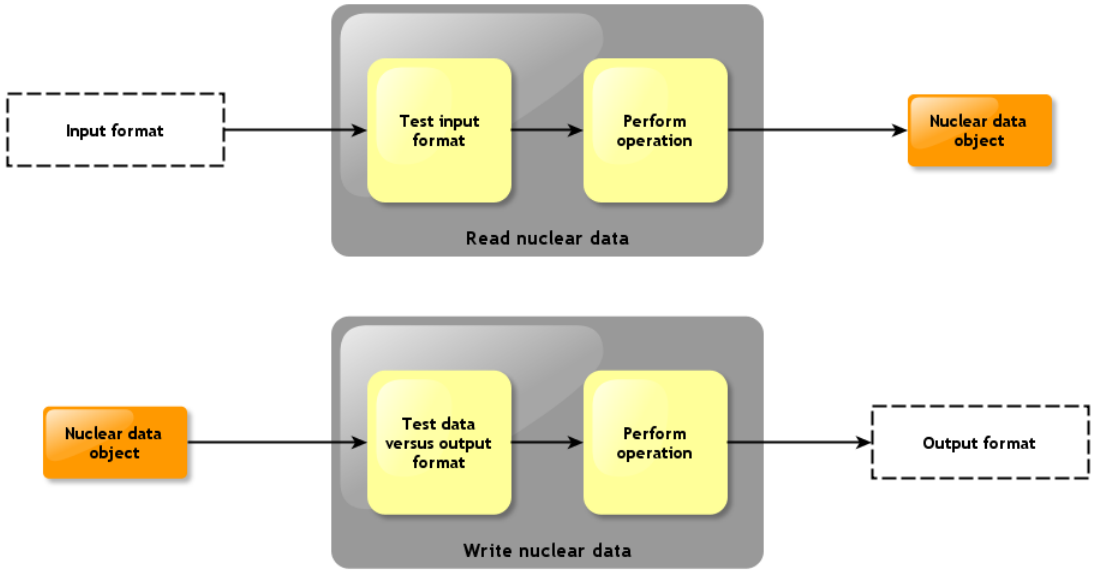
Design example

- The high level nuclear data object can be used to represent data in several representation types
  - For example: an angular distribution can be Legendre polynomials, an analytical function or a tabulated angular distribution
  - Metadata identifies the data representation type
  - The various components do not have functions to read/write to files



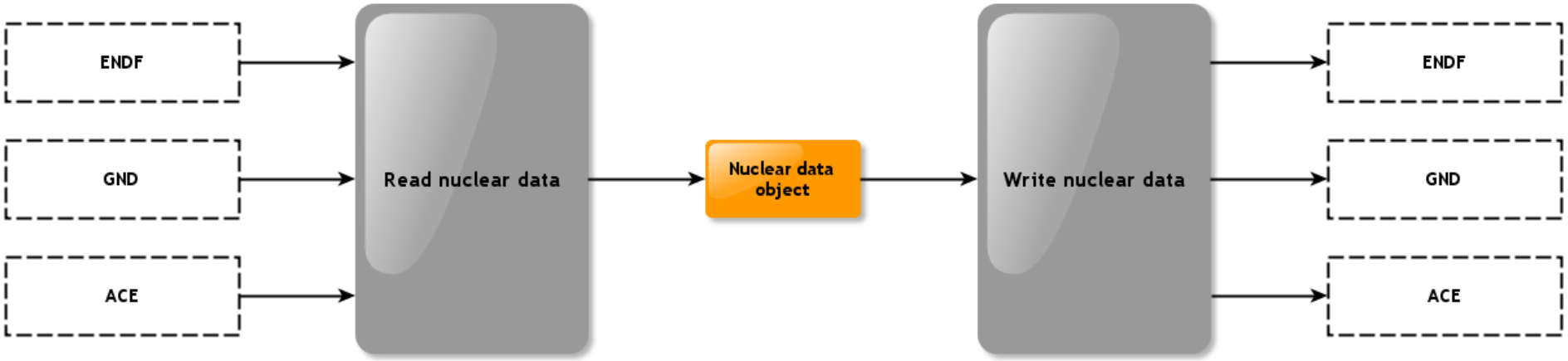
Design example

- Reading and writing to a given format is a generic operation
  - The data object only knows “what” it is, not “where” it comes from
  - These operations can be implemented for any format (essentially a low-level API for each format)
  - Compatibility testing can be done using the metadata



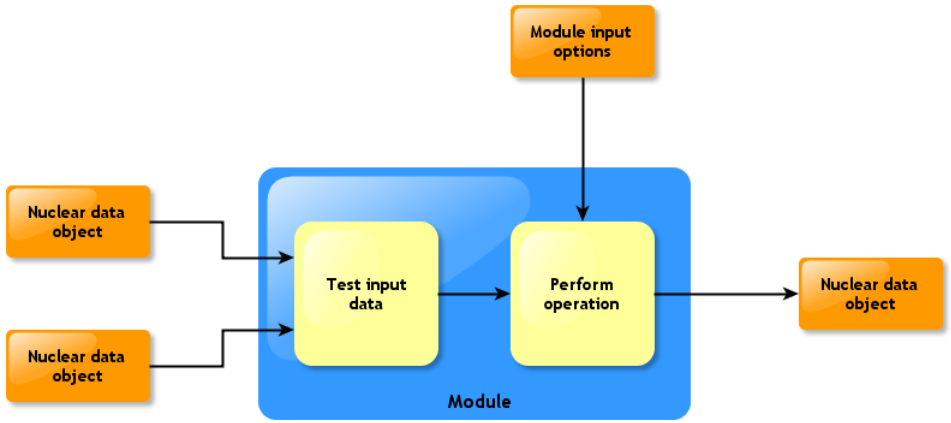
Design example

- With this approach one can imagine the following operation
  - Implement readers and writers for each format type
- Compatibility issues are dealt with using the metadata
  - An unprocessed ENDF file cannot be transformed into an ACE file



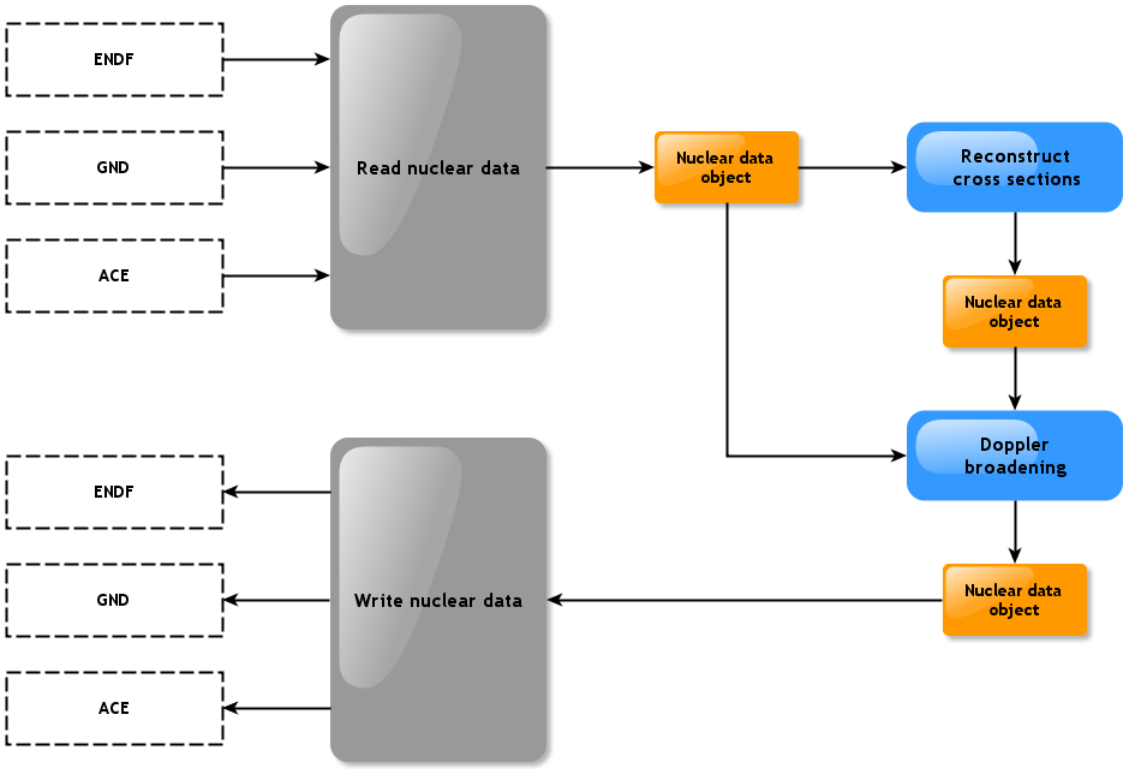
Design example

- Data manipulation and processing is also a generic operation
  - Manipulation of nuclear data consists of changing its representation type
    - For example: cross section reconstruction and linearisation
  - Before an operation can be performed, the module needs to test whether or not it can perform the operation
    - For example: cross section data needs to be linearised for a basic Doppler broadening operation



Design example

- Putting it all together: a basic processing sequence



Organising the work

- Constraints and requirements
  - The design of the new high level structure has to be rather advanced
    - It is easier to change a conceptual design document
  - Be clear on what we want from the beginning
- Milestones
  - Conceptual design of the API
    - Nuclear data object
    - Operations: reading, writing, modules and module types
  - Implement the nuclear data object and its components
  - Build the generic framework for reading and writing to file formats
  - Implement reading from a file
  - Implement writing to a file
  - Think about processing

Conclusions

- First of all: this is my take on the topic
- General decisions are to be made
  - What to store and which representation types (structure)
  - Which formats to support (ENDF, new format, GND, etc.)
  - Where does the API end and where does processing begin?
- IRSN is probably going into this direction with its software
  - Work is starting on a software requirements and conceptual design document for our GAIA software
  - This is compatible with some of the needs of the community