Supporting Generic Programming in a Multi-Language Component-Based Environment

Wael R. Elwasif *, Gregory Brown †, Thomas C. Schulthess *, David E. Bernholdt *

{elwasifwr, browngp, schulhesstc, bernholdtde}@ornl.gov

* Computer Science & Mathematics Division
  Oak Ridge National Laboratory

† School of Computational Science and Information Technology
  Florida State University
Overview
We examine some of the issues and possible solutions involved in supporting generic programming in a multi-language component-based environment which also includes languages that do not support generic programming.

This work stems from an effort to make the Ψ−Mag toolkit available in componentized form. The Ψ−Mag toolkit is a C++ library for computational materials sciences that makes extensive use of generic programming principles through the use of C++ templates. Our goal is to export the functionality of the Ψ−Mag toolkit in componentized form that follows the specification of the Common Component Architecture (CCA). The CCA is an open component environment specifically designed to support high-performance scientific computing.

Accommodating components written in the most commonly used HPC programming languages is one of the underlying design goals of the CCA. Language interoperability in CCA is achieved through the use of the Scientific Interface Definition Language (SIDL) and its associated C++ Compiler, and the SIDL object model that is closely related to the Java programming language, which has no support for template-based generic programming. This object model is mapped into implementation objects in the most commonly used HPC languages (C, C++, Fortran, and Python) via the C++ compiler.

In this poster, we present preliminary results in exporting the functionality embodied in the Ψ−Mag C++ template-based Ψ−Mag toolkit libraries to other languages using the SIDL/Babel language interoperability tools. Early results suggest that with proper design of the SIDL object layer that wraps Ψ−Mag objects, the library can be used in a multi-language environment without the need to alter library internals.

Goals
This work aims at exporting the functionality embodied in the Ψ−Mag toolkit in componentized form that is compliant with the CCA specification. Towards that end, we address the general problem of accessing C++ template-based generic code in a multi-language environment using the SIDL/Babel language interoperability tools. We work to understand the boundaries of the Ψ−Mag object model to seek design patterns and programming techniques that facilitate such access without the need for intrusive modifications to the underlying generic library design and/or code. We look for a combination of manual and automated solution that simplify the process whenever possible.

Technical Issues
Support for generic programming in a multi-language component-based environment is particularly challenging for two reasons. First, many languages do not support generic programming. While the language interoperability layer can be used in many cases to augment the target language, for example, by providing an object model to non-oriented languages, generic programming requires more extensive modification to the base language. Second, Ψ−Mag’s compile-time mechanism for resolving templates based on how they are used is at odds with the template design patterns that need to be used by the library to implement the interface. This latter reason is further broken down into two main issues, type mapping and object instantiation. The type mapping component identifies the types used by the class, this information is used to instantiate a version of the class template that uses those concrete types.

The Ψ−Mag Tool Set
The Ψ−Mag tool set is a C++ library for computational magnetism from which scientific applications can be efficiently built. Using generic programming techniques and C++ templates, it provides a wide range of generic libraries, e.g. special functions, integration, and random number generators. It also provides tools specific to computational magnetism, from methods for calculating the energies and fields of particular configurations of the magnetization to the Fast Multipole Method for calculating dipole-dipole interactions. This tool set is managed by UT-Battelle, LLC for the US Dept. of Energy under contract DE-AC-05-00OR22725.

Approach
Multiple SIDL interfaces are instantiated that support all possible types used in a generic Ψ−Mag interface.

- A single SIDL class implements all possible interfaces, using a factory design pattern to select which interface (and underlying Ψ−Mag class) is instantiated.
- SIDL objects wrap underlying Ψ−Mag objects.
- SIDL Method arguments are wrapped on the fly into Ψ−Mag compliant interfaces that are passed down to native Ψ−Mag code (this allows Ψ−Mag to be extended by code written in other languages).

Implementing the SIDL wrapping object layer

References