Grand challenge computing applications often require that multiple software components be coupled together in a variety of configurations on parallel computing platforms, utilizing a mix of data and task parallelism. The components in an application may run sequentially, concurrently, or in some combination of these modes, and the interactions necessary to connect components may involve data redistribution, spectral or grid transformations, time averaging, unit conversions, etc. Climate, weather, and related geophysical applications fit this profile, as do many signal processing applications. Design goals for such systems include the ability to use the same component in multiple contexts, to swap different implementations of a component into an application, and to assemble and extend coupled systems easily; in short, reuse and interoperability.

A recognized design pattern for achieving these goals is the mediator [1], in which one object encapsulates how a set of other objects interact. The mediator serves as an intermediary, and keeps objects from referring to each other explicitly. Mediators are called couplers in climate and related sciences [2,3], conduits in the Parallel Vector Library (PVL) [4], and other names (hubs, directors, etc.), depending on the domain. The great advantage of this pattern is the overall simplification of interactions. The great disadvantage is the following, from Gamma et al.:

“The Mediator pattern trades complexity of interaction for complexity in the mediator. Because a mediator encapsulates protocols, it can become more complex than any individual colleague. This can make the mediator itself a monolith that’s hard to maintain.”

In the remainder of this document we look at approaches that the Earth System Modeling Framework (ESMF) [5,6] has adopted in order to simplify its mediator/coupler objects, while retaining flexibility of configuration and performance.

ESMF provides abstractions for components and couplers, an architecture for their assembly, and a set of utility and data classes for component composition. ESMF is the technical foundation of the cross-service DoD Battlespace Environments Institute, the NASA Modeling Analysis and Prediction Program for Climate Variability and Change, the Community Sediment Transport Model, and many individual modeling systems. The current list of ESMF components includes models representing global physical domains (e.g., atmosphere, ocean), models representing regional domains and specific processes (e.g., near-shore ocean, radiation), and computational functions such as data assimilation. Although ESMF is targeted for the Earth and space domains, and its data structures reflect the grids and transformations particular to those sciences, the architectural approach is quite applicable to other domains. In fact, a number of the strategies the ESMF team has adopted are drawn from other domains and general computational frameworks.

These strategies are outlined below.

1. **Construction of the mediator using the same standard interfaces as the components that it connects.** In ESMF the same code base and interfaces are used to wrap both scientific components (GridComp objects) and couplers (CplComp objects). The different names of these objects are simply a convenience for users. Both types of components have Initialize, Run, and Finalize methods with standard interfaces. The argument list for each method is straightforward and includes an import State containing data in, an export State containing data out, and a Clock. The Clock stores start time, stop time, and timestep information, and may also hold a set of Alarms to identify periodic or unique events. Users customize a component by attaching their own routines to the component’s standard methods, and then assemble a collection of components into a single executable. The ESMF team has found that this approach enables users to quickly understand an application’s architecture, and what coupler inputs and outputs are. Further, user customization of couplers encourages simplicity by avoiding the need for a generic coupler that can handle many different scenarios.

2. **Hierarchical construction of the data passed into and out of mediators.** ESMF State objects can contain ESMF Field, Bundle (of Field), and Array types, but they can also contain other State objects. This enables, for example, a set of ocean, land, and atmosphere States to be nested within a composite State, and to be passed into a coupler for an operation in which boundaries between ocean and land must be reconciled before the surface can interact with the atmosphere. To preserve performance,
States can reference user data. The hierarchy of States is a mechanism that allows for data organization, simple standard interfaces, and flexibility of configuration.

3. **Hierarchical construction of the mediator, so that coupling occurs at multiple levels.** This aspect of the ESMF architecture is essential for many-component applications, and it is the feature that most strongly affects complexity of couplers. Components can contain other components, so that an ESMF application consists of an extensible tree of scientific components and couplers. Complexity of a central coupler is reduced by splitting its function among multiple couplers, which can operate at different scales. For example, Figure 1 shows the structure of the GEOS-5 Atmospheric General Circulation Model [7] developed at NASA Goddard. Each box in the diagram is an ESMF component, and the components are structured hierarchically and coupled at their local scope. Couplers in this case are sufficiently simple that the Goddard group has developed an auto-generation tool to write them.

4. **Decoupling of the communication mechanism from the system architecture.** The standard ESMF paradigm allows only intra-component data communications. This means that a coupler component must be defined on the union of the computational resources (i.e., processors) of the components that it couples. Scientific components make an on-resource call to a coupler when they want to communicate. This rule was suggested by the Common Component Architecture (CCA) project [8]. The ESMF team has found that it simplifies control syntax and communications without much limiting the varieties of data flow that ESMF can support, or its performance.

5. **The provision of regridding and redistribution tools that encapsulate many of the transformations within mediators.** Insofar as possible, ESMF encourages the internals of couplers to be written using ESMF utilities. These tools take as arguments the same data structures (States, Fields, Bundles, Arrays) that are used at the component and coupling level. This approach is drawn from comprehensive frameworks such as the Space-Time Adaptive Processing Library (STAPL) from Lincoln Laboratory [9]. The idea is to minimize the lines of code that are required to write a coupler, as well as the number of packages that are required to do so.

We note an additional strategy that is being developed by a group at the University of Maryland in collaboration with the ESMF team. This group is implementing ESMF State put/get calls that enable from anywhere/to anywhere data communications between models and keep communicating models in separate executables. The intent is to address coupling problems that cannot be easily handled through the hierarchical constructs described above - for example, a coupling of two monolithic models that require a data exchange mid-timestep, deep within their call trees. The preference in this case is to introduce an entirely separate mechanism to support these “special case” couplings rather than extend the standard paradigm.

The design of mediator objects is difficult to do well, and especially so for many-component systems. As a measure of its success in this area ESMF can show growing acceptance in the Earth science community. However, the ESMF team is indebted to lessons learned and innovations from efforts in other domains, and one of the ESMF team’s most valuable lessons learned is to continue to seek out cross-disciplinary interactions for the insights they bring to common software problems.

**Acknowledgements**

This work is supported by the Department of Defense, the National Science Foundation, and the National Aeronautics and Space Administration. Thanks to Julien Chastang and the members of the ESMF Core and Joint Specification Teams for many helpful conversations.

**References**

[1] Gamma, E., R. Helm, R. Johnson, and J. Vlissides, 1995: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley.


