Managing Software Complexity and Variability in Coupled Climate Models

Spencer Rugaber, Rocky Dunlap, Leo Mark, Sameer Ansari
College of Computing, Georgia Institute of Technology

Modern coupled climate modeling software is complex, supporting multiple modeling domains, grids, numerical methods and interprocess communication mechanisms in scientifically rigorous ways to predict future climate. However, this complexity hides a rich design space that scientists can exploit to produce a wide variety of simulation systems. In this article, we describe the complications that arise from such diversity and techniques that can be used to control it. In particular, we describe a detailed feature analysis we have conducted of the climate modeling technology design space and present two example applications that demonstrate how the complexity can be controlled.

Coupled Climate Models

Climate models are complex simulation programs used to predict future climate. In a climate model, the Earth is partitioned into *cells* with various associated data items (*fields*), such as temperature and wind speed. At every simulated time step, data from a given cell propagates to neighboring cells in a way that conserves physical properties such as total energy. The partition is called a *grid*, and it may be two or three dimensional, plus a temporal dimension. A variety of different *domains* have been modeled, such as the Earth's atmosphere, ocean, land surface and cryosphere.

It has been recognized for some time that integrating two or more models taken from different domains can yield a more credible estimate of future climate than can a single model. When this is done, the models are said to be *coupled*, and a *coupler* component is responsible for the transfer of data across domain boundaries. Although simple in principle, couplers are complex software components. Among their responsibilities are transferring field data, which may involve data type and unit conversion, ensuring that conservation laws are upheld, coordinating different time step lengths, interpolating data when different grid resolutions and topologies are used, and supporting computations divided across multiple processors and address spaces.

Coupled climate models are *variation-intensive* software systems. The software must be flexible enough to handle experimental variation—that is, to enable sensitivity analyses through ensembles of related runs. The software may also provide the experimenter with a set of diverse scientific components that allow the user to test the effects of different physical parameterizations on climate. Additionally, variation at the architectural level allows software engineers to take advantage of different parallel computing and memory configurations.

The software engineering community has developed analysis and implementation techniques for dealing with variation-intensive software. *Feature analysis* is one such technique that we have employed to better understand the climate-coupling problem. Our feature analysis has pointed out many variations on the underlying theme of coupling. We have identified the main design dimensions and determined how effectively existing technologies address the dimensions. We first describe the dimensions and then give two examples of applications that our feature analysis

enables: automatic generation of couplers and automatic configuration of coupled models on the cloud. We conclude by discussing some of the issues that our analysis has raised, both for the climate modeling community and for software engineers.

An Example Coupled Model

The Community Earth System Model (CESM) is an example of a complex coupled climate model [1]. It was developed at the National Center for Atmospheric Research (NCAR) to study climate change and is a participant in an ambitious set of climate experiments to be featured in the Fifth Assessment Report (AR5) of the Intergovernmental Panel on Climate Change (IPCC), scheduled for release in 2013–14. It is designed to be configured in many different ways to support a wide range of scientific requirements. However, specifying all possible parameters and configuration options is laborious and time-consuming for users [2].

The CESM 1.0 distribution contains roughly 800 files consisting of over half a million lines of code, 45,000 lines of configuration scripts, and multi-gigabyte input datasets. The models directory contains 22 different *components* (single-domain models), including atmosphere, land, sea-ice, ocean and a coupler that defines boundary layer interactions between the components, all written primarily in Fortran 90.

The advantage of a coupled model framework like CESM is that it allows the simulation software to be divided into components that can be modified or replaced individually. Taking into account only supported component sets and grids, there are 1,132 different total configurations possible, and this does not include modification of configuration files, which have a large range of possible values, placing the possible number of different configurations in the hundreds of thousands. As CESM continues to grow, it is increasingly important to not only understand how a configuration is created, but also what are the relationships and constraints introduced by its feature.

Understanding Coupling Complexity with Feature Analysis

To better understand the nature of coupling in climate models, we have applied a software engineering technique called *feature analysis*. Feature analysis was developed to support software product-line engineering, in which a set of related software products is characterized in terms of their mandatory and optional features [3]. One product of feature analysis is a set of feature diagrams.

We applied feature analysis to determine the software engineering aspects of six technologies that have been developed to support coupling of climate models. Overall, the feature model we produced comprises eleven diagrams and 200 total features [4]. It paints a picture of a design space rich in ways to elaborate on the basic theme of coupling. Before giving an example of a resulting feature diagram, we first describe the technologies that we studied, which range from traditional support libraries, through comprehensive frameworks to an automated coupler generator.

Typed Data Transfer (TDT) [5] is a lightweight communication layer that abstracts several communication mechanisms into a single application programming interface (API). XML configuration files describe the mechanisms and the data types to be transferred. Calls to simple read and write routines are inserted where data is required or produced, respectively. Changing the underlying mechanism requires no changes to the code itself, only to the configuration file.

Model Coupling Toolkit (MCT) is a toolbox for building couplers, including a module for describing domain decomposition, a random-access storage type for field data, communication schedulers for parallel repartitioning of distributed arrays, grid-to-grid interpolation routines, a physical-space representation for storing grid point details, a utility for spatial integration, accumulators for temporal summation and averaging, and a merge facility for combining multiple data sources [6].

The Earth System Modeling Framework (ESMF) is a coupling framework that supports constructing coupled models from components [7]. ESMF provides a set of technical services (*infrastructure*) and abstract component interfaces (*superstructure*). A large number of technical services are provided by ESMF including domain decomposition, repartitioning, interpolation, scatter/gather of field data, intermodel time coordination, support for different calendars, tools for configuration management, and the ability to output field-level metadata.

The **Flexible Modeling System (FMS)** [8] is another component-based coupling framework offering both an infrastructure layer of common technical services and an superstructure layer for defining top level structures in the coupled model. The technical services provided by FMS include I/O, exception handling, and functions for interpolation and repartitioning of field data in parallel. Whereas ESMF defines generic component interfaces, FMS provides domain-specific scientific interfaces for a pre-determined set of components (atmosphere, ocean, ocean surface/sea ice, and land surface models). The scientific interfaces are defined both in terms of a set of control subroutines and specific data structures for holding fields exchanged between models.

OASIS/PSMILe is a complete implementation of a transformation and interpolation engine and associated driver [9]. The Driver–Transformer (coupler) and constituent models remain as separate executables during a model run. Communication with the coupler is accomplished by inserting API calls and linking the PSMILe library to each constituent model. The configuration of the coupled application is described in XML files including, for each constituent model, a description of the source and target of input or output fields, the exchange frequency, and the transformations that should be applied.

The **Bespoke Framework Generator (BFG)** [10] is designed to enable flexibility in configuring and deploying instances of coupled models. It is a framework generator because it produces customized packaging and control code based on user-supplied metadata [11]. The framework code generated by BFG2 invokes component models and enables them to communicate. A design constraint for BFG2 is the desire to leave component models completely unchanged thereby precluding re-architecting of model code to match predefined interfaces, inserting calls to specialized functions for sending or receiving data, or even adding annotations at potential communication points in model code. The user provides configuration metadata to the BFG2 code generator using XML files. Unlike the other coupling technologies analyzed except TDT, BFG2 does not have knowledge of the numerical properties of the constituent models other than the number and size of array dimensions and does not support utility functions such as repartitioning or interpolation natively.

An Example Feature

Figure 1 is one of these diagrams produced by our feature analysis, describing a feature called Driving. In the figure, nodes, corresponding to features, are connected with directed edges, and edges have decorations that define the semantic relationship between parent and child nodes.

Some features are mandatory (denoted by an edge ending with a filled circle), and others are optional (denoted by an edge ending with an open circle). If the edges connecting a set of feature nodes to their parent are connected with an arc, then those features are alternatives, requiring exactly one to be selected. If the arc is filled in, one or more may be chosen. Finally, features can be connected (non-hierarchically) by a labeled dashed lines, indicating an interfeature constraint.

The *Driving* feature has to do with how the coupled models are coordinated and how they are stepped forward in time. In the figure, the Startup Extent subfeature shows that in some cases the user is responsible for starting not only the driver but also the constituent models while, in others, the driver takes on the responsibility of starting the constituent models. Note that there is a *requires* constraint between the Startup Extent/Just Driver and Location of Driving Code/Driver subfeatures because the driving code must appear in a separate Driver if the Just Driver subfeature is selected.

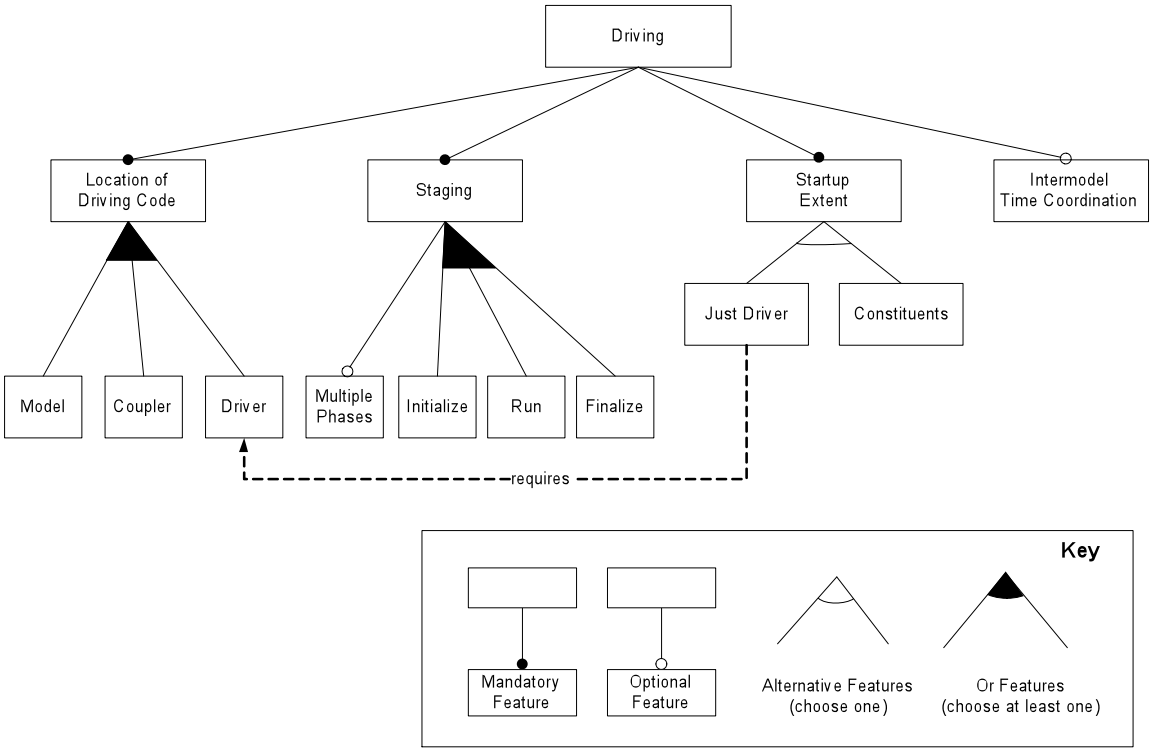

Figure 1 Driving Feature

Our feature analysis provides a scaffold against which the complexity of coupled climate models can be understood. Importantly, it indicates how the multifarious responsibilities of such models can be architecturally realized in a variety of ways, for example with either centralized (OASIS) or architecturally distributed couplers (ESMF), with model code invoking support technology (TDT, MCT, OASIS) or being invoked by it (ESMF, FMS, BFG), and with the technology providing explicit intermodel time coordination mechanisms (OASIS, ESMF, FMS, BFG) or leaving this task

to the user (TDT, MCT). This understanding can be in exploited in a variety of ways, two of which we illustrate in the next section.

Applications of Feature Analysis

The feature analyses we performed taught us a great deal about configuration of climate models and variations in the design of couplers. We now illustrate how this knowledge might be used by discussing two prototype applications: a coupler generator and an automated configurator of coupled climate models on the cloud.

Automatic Generation of Couplers

One way to deal with complex software is to automatically generate part of it from a high-level, declarative description. Generation not only reduces the amount of source code that has to be written, but also more closely expresses user-selectable variation in terms understandable by the user. Feature analysis supports this approach by enabling users to express their choice of features in the form of a product configuration that is, in turn, used to automatically generate the desired product. We have made use of our analysis of coupling technologies to build a prototype ESMF coupler generator, called *Cupid*. Figure 2 portrays its conceptual architecture.

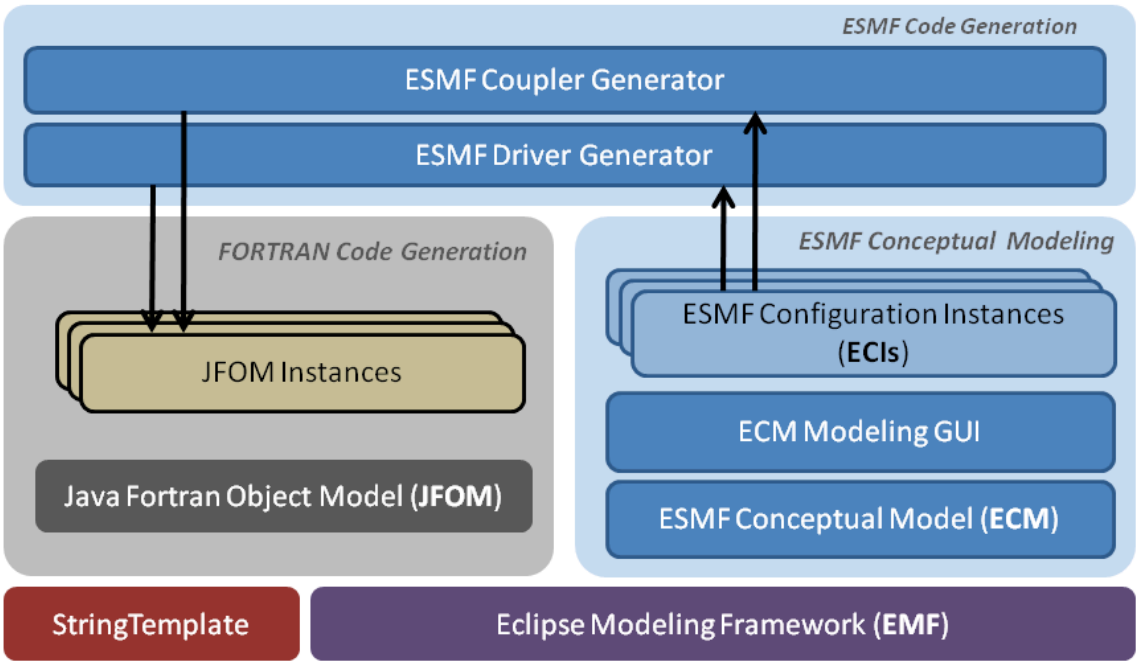

Figure 2 Cupid Architecture

Cupid includes a set of Java classes called the ESMF Conceptual Model (ECM) that represent ESMF API elements. Instances of the ECM represent particular coupling configurations, selected from the many options determined by our feature analysis, which serve as input to the coupler and driver code generators. The generators do not output source code directly; rather, they

produce an intermediate form called the Java Fortran Object Model (JFOM). JFOM instances are then translated into textual Fortran source code using a string template library.

The code generated by Cupid successfully reproduces the coupler and driver code in two system tests that ship with the latest distribution of ESMF. The code can be compiled with the system tests and works the same as when using the original, hand-written couplers and drivers. The Cupid proof of concept shows the feasibility of generating coupling code that targets one specific coupling technology—ESMF. Currently, the generated couplers deal with a small number of coupling fields and the coupled models are stubs, not implementing any real science. The system must be expanded to target the more complex models in operation today.

Configuration of CESM for the Cloud

We have made use of our feature analysis in a second way—for configuring and executing machine images for coupled model runs on a cloud-computing platform. Cloud computing provides an alternative to the currently dominant practice of running climate simulations on dedicated super computers. Running simulation experiments on the cloud increases accessibility and flexibility with only acceptable reductions in performance [12].

In order to take advantage of the cloud, two sources of complexity must be overcome: configuring the simulation and managing execution. As the description of CESM given above indicates, a scientist has to choose from the many component models and grids in such a way that no scientific or technical constraints are violated. Managing execution is also complex, requiring selection of suitably powerful machine resources, creation of machine images for them, populating the images with the required software, building any artifacts that are not already part of the images, deploying the resulting images, downloading the necessary input data, and, finally, initiating model execution.

We have built a prototype tool called the CESM Cloud Configurator (C3) whose goal is to simplify CESM configuration and execution management by providing an abstracted interface oriented towards scientists instead of software engineers. We translated our feature model into an OWL[1] ontology (a standard knowledge representation language) using Protégé[2], an ontology-editing tool. Using an OWL ontology provides a framework for automated reasoning tools to infer relationships between features, such as when a configuration is invalid or when two feature choices are mutually exclusive. From this, the user can choose a subset of options and the rest will be inferred from the ontology. The Protégé also supports logical subsumption, that is, it knows when one set of configuration choices is a superset of another. This feature avoids duplicate effort, such as when an existing saved machine image can be adapted more easily than a new machine image can be generated. The resulting choices are saved into an XML file.

The current version of C3 is able to instantiate a skeleton machine image containing CESM. C3 reads in the saved XML specification file, configures a cloud instance, and starts the simulation. A management window is also provided, permitting the user to interact with the instance remotely.

[1] http://www.w3.org/TR/owl2-overview/

[2] http://protege.stanford.edu/overview/

Looking Ahead: Systematic Variation in Climate Models

Our two example applications illustrate how a systematic understanding of the design space of couplers can be used to advantage. However, the space of possible coupled climate models is large and assembling one that is both scientifically meaningful and efficient, is difficult. To support the process of assembling individual components into fully coupled climate models, a variety of issues remain to be addressed.

Improvements should be made in how software variability is implemented in climate models. We have already seen how state-of-the-art climate models require scientific, numerical, and architectural variability. However, for reasons of expediency and the need to retain backward compatibility with legacy codes, these variation points are often implemented in an ad hoc manner, resulting in highly complex code with many interdependencies. This further hinders the ability of model developers to fully understand an entire climate model codebase—which, according to Randall, is an arguably impossible task for a single individual [13]. Further, code complexity can be a significant barrier to verification and validation of climate models. Ultimately, we seek more *intentional* climate models—that is, software that is a more direct encoding of what model developers intend [14]. A potential solution is to improve the modularity of existing codes by organizing it around features. Organizing a program into well-designed modules can promote intentionality when the modules have a clear correspondence with domain-level concepts.

Although modularity in software systems is linked to a number of advantages, there are performance costs involved with modular designs due to potential efficiency loses at module interfaces. The antagonistic relationship between modularity and performance has been recognized as a general principle in many kinds of engineering systems. For example, the findings of Holtta et al. suggest that integrated, tightly coupled architectures are more likely to favor increased technical performance than loosely coupled, modular architectures [15]. On the other hand, modular architectures offer better support for other qualities, such as flexibility, interoperability, robustness, and the ability to distribute work among members of a software team. Striking a balance between modularity and performance in climate models requires careful consideration of the tradeoffs involved.

Coupled climate models are complex, but that complexity can be controlled if a systematic characterization of their variability, such as that provided by a feature model, is available. Moreover, such a characterization can be leveraged in a variety of ways such as in support of automatic generation of couplers and configuration of execution images. Most importantly, managed control of this complexity is essential to support future scientific advances, in which more models are added and model interdependencies increase.

Acknowledgements
This work is supported by grants from the National Science Foundation: the Earth System Curator project (#0513635) project and a NSF Graduate Research Fellowship (#0513762). Special thanks to those who provided personal correspondence regarding coupling technologies including Sophie Valcke, V. Balaji, Tony Craig, Mariana Vertenstein, Rob Jacob, Jay Larson, Cecelia DeLuca, Bob Oehmke, Ryan O'Kuinghttons, Rupert Ford, Graham Riley, and Rene Redler.

References

[1] M. Vertenstein, T. Craig, A. Middleton, D. Feddema, and C. Fischer. *CCSM 4.0 User's Guide*. 2010, http://www.cesm.ucar.edu/models/ccsm4.0/ccsm_doc/book1.html.

[2] S. M. Easterbrook and T. Johns. "Engineering the Software for Understanding Climate Change". *IEEE Computing in Science and Engineering,* 11(6):65-74, 2009.

[3] K. Czarnecki and U. W. Eisenecker, *Generative Programming: Methods, Tools, and Applications*: Addison-Wesley, 2000.

[4] R. Dunlap, S. Rugaber, and L. Mark. "A Feature Model of Coupling Technologies for Earth System Models." Georgia Institute of Technology, GT-10-18, 2010.

[5] C. Linstead. *Typed Data Transfer (TDT) User's Guide.* Potsdam Institute for Climate Impact Research, 2004, p. 21.

[6] J. Larson, R. Jacob, and E. Ong, "The Model Coupling Toolkit: A New Fortran 90 Toolkit for Building Multiphysics Parallel Coupled Models," *International Journal for High Performance Computing Applications.* 19:277-292, 2005.

[7] V. Balaji, B. Boville, S. Cheung, N. Collins, T. Craig, C. Cruz, A. d. Silva, C. DeLuca, R. d. Fainchtein, B. Eaton, B. Hallberg, T. Henderson, C. Hill, M. Iredell, R. Jacob, P. Jones, E. Kluzek, B. Kauffman, J. Larson, P. Li, F. Liu, J. Michalakes, S. Murphy, D. Neckels, R. O. Kuinghttons, B. Oehmke, C. Panaccione, J. Rosinski, W. Sawyer, E. Schwab, S. Smithline, W. Spector, D. Stark, M. Suarez, S. Swift, G. Theurich, A. Trayanov, S. Vasquez, J. Wolfe, W. Yang, M. Young, and L. Zaslavsky. "ESMF User Guide Version 3.1." 2009.

[8] V. Balaji. "Flexible Modeling System." *The FMS Manual: A Developer's Guide to the GFDL Flexible Modeling System.* Princeton, NJ, 2002.

[9] R. Redler, S. Valcke, and H. Ritzdorf. "OASIS4—A Coupling Software for Next Generation Earth System Modeling." *Geoscientific Model Development,* 3:87-104, 2010.

[10] C. W. Armstrong, R. W. Ford, and G. D. Riley. "Coupling Integrated Earth System Model Components with BFG2." *Concurrency and Computation: Practice and Experience,* 21:767-791, 2009.

[11] B. Clifford, I. Foster, J.-S. Voeckler, M. Wilde, and Y. Zhao. "Tracking Provenance in a Virtual Data Grid." *Concurrency and Computation: Practice and Experience,* 20:565-575, 2008.

[12] C. Evangelinos and C. N. Hill. "Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2." *Cloud Computing and Its Applications,* 2008.

[13] D. Randall. "The Evolution of Complexity in General Circulation Models." *The Development of Atmospheric General Circulation Models.* L. Donner, et al., eds., Cambridge University Press, 2011, p. 272.

[14] C. Simonyi, M. Christerson, and S. Clifford. "Intentional Software." *Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '06)*. Portland, Oregon, 2006.

[15] K. Holtta, E. S. Suh, and O. de Weck. "Trade-off Between Modularity and Performance for Engineered Systems and Products." *International Conference on Engineering Design,* Melbourne, 2005.

Sidebar - Acronym List

API - Application Programming Interface
AR5 - Fifth Assessment Report from the Intergovernmental Panel on Climate Change
BFG - Bespoke Framework Generator
CESM - Community Earth System Model
CAM - Community Atmosphere Model
C3 - CESM Cloud Configurator
ECM - ESMF Conceptual Model
ESMF - Earth System Modeling Framework
FMS - Flexible Modeling System
IPCC - Intergovernmental Panel on Climate Change
JFOM - Java Fortran Object Model
MCT - Model Coupling Toolkit
MPI - Message Passing Interface
NCAR - National Center for Atmospheric Research
OASIS/PSMILe - Ocean Atmosphere Sea Ice Soil Coupler with PRISM System Model Interface Library
OWL - Web Ontology Language
TDT - Typed Data Transfer
XML - Extensible Marking Language