# Table of Contents

1. **Scientific/Technical Management**
   - 1.1 Objectives and Expected Significance
     - 1.1.1 Objectives
     - 1.1.2 Expected Significance
   - 1.2 Technical Approach and Methodology
     - 1.2.1 Testing
   - 1.3 Perceived Impact to State of Knowledge
   - 1.4 Work Plan
     - 1.4.1 Key Milestones
     - 1.4.2 Management Structure
     - 1.4.3 Contributions of Principal Investigator and Key Personnel
     - 1.4.4 Collaborators and Consultants
     - 1.4.5 Risk Management
   - 1.5 Management of Source Code and Information

2. **References and Citations**

3. **Biographical Sketches**

4. **Current and Pending Support**
   - 4.1 Current Awards
   - 4.2 Pending Awards

5. **Budget Justification**
   - 5.1 University Of Colorado Budget Narrative
   - 5.2 University of Colorado Budget Details
   - 5.3 Georgia Institute of Technology Budget Narrative and Details
   - 5.4 NASA Budget Narrative
   - 5.5 NASA Budget Details

References and Citations
1 Scientific/Technical Management

The complexity of Earth system models makes code additions and modifications difficult and time consuming. We propose a strategy that will bring modern software engineering tools to bear on the problem, and will introduce a development environment for modeling that can be used in classroom or training settings, and to automate aspects of code modification for a model development team at NASA.

We will work along two convergent paths: development of the Cupid Integrated Development Environment (IDE), a suite of user-friendly tools for creating, modifying, and running Earth System Modeling Framework (ESMF)-based applications; and integration of the NASA GISS Model E with ESMF infrastructure. The project goals will be to arrive at a Model E with better parallel performance, modularity, and ease of use, and to enable that model to be used in teaching and training exercises within the Cupid IDE.

Cupid – software that facilitates coupling – will leverage the existing NASA MAP Modeling Environment (MAPME) Workflow Engine to perform job submission and monitoring. The MAPME team has already resolved thorny issues related to automated workflows executing in the high security setting of a NASA computing center. The Cupid system will initially target users of Model E working on NASA computers. However, it will be designed with an eye toward how it can be extended for other ESMF-compliant models, such as the GEOS-5 model, and how it can be run on computers outside of NASA.

The Cupid IDE will advance the state of the art in Earth system model software development, combining proven approaches with modern technologies. Its design is informed by a number of end-to-end modeling systems that, in their own ways, simplify or automate running complex Earth system models. Examples include the GFDL Flexible Modeling System Runtime Environment (FRE)\(^1\) and the Community Climate System Modeling (CCSM) Portal developed at Purdue University\(^2\). Since its introduction in 2003, GFDL modelers have come to rely on FRE, which is described as “a completely indispensable element in our testing, and a successful element in long production runs where a sustained workflow was important.”\(^3\) The CCSM portal has likewise become a valued teaching tool at Purdue, where it enables students to gain hands-on experience with running a climate model, with few technical prerequisites. The design of these software tools reflects the requirements of their respective environments. FRE must give researchers free rein to modify the GFDL models, and so accommodates extensive changes and configuration options. The FRE user interaction involves editing an XML file that fully specifies the run, an approach whose explicitness is likely reassuring to senior scientists but intimidating to students. The CCSM portal, on the other hand, offers students a simple GUI interface, designed around a set of standard runs, with somewhat limited configuration options.

Integrated Development Environments (IDEs), many of which are extensions of the Eclipse\(^4\) toolkit, are exciting because they have the potential to offer both flexibility in the kinds of additions and modifications that can be made to a modeling system, and a user interface that is appealing, intuitive, and interactive. IDEs typically support all stages of the software process, including writing code, testing, debugging, and deployment to the production environment. Tools are also emerging that address high-performance software, including debugging and monitoring execution of distributed memory applications\(^5\). We anticipate that the IDE-based approach of Cupid may be suitable for a research environment like GISS as well as teaching and training scenarios where more in-depth code modifications are desired. Since established model users may find it hard to adopt new practices, our user interactions will focus mainly on students
and trainees. The proposed work will be excellent preparation for a modeling course based on Model E.

An early prototype of Cupid, based on the Eclipse Modeling Framework\textsuperscript{6} was created under the NSF-funded Curator project\textsuperscript{7}. This proof-of-concept software demonstrates that Cupid can support the creation and manipulation of ESMF components. The ESMF data structures and constraints, especially at the level of large components (atmosphere, ocean, sea ice, etc.) give the IDE a domain-specific feel and help the modeler to architect the overall modeling application. Future work will include incorporating ESMF “usability layers” into Cupid. A \emph{usability layer}, which is built using a subset of ESMF calls, imposes usage, metadata, and other conventions on the model developer in order to constrain the implementation of components and improve interoperability. The two main usability layers are MAPL, developed at NASA Goddard and the National Unified Operational Prediction Capability (NUOPC) Layer, developed by a Navy/NOAA collaboration. This aspect of our work will be exploratory, and the relation with GISS will depend on the extent to which Model E scientists wish to balance the adoption of conventions with the ability to freely structure code.

Functional goals of incorporating ESMF infrastructure in Model E at the large component level include improving aspects of parallelization and performance of the model, particularly with respect to grid transformations and data communications; improving modularity and thus the ability to modify and test code in a controlled, limited way; easing the introduction of new ocean and ice models; and introducing automated code documentation. Whether to go further with componentization (e.g., for wrapping atmospheric dynamical cores) will need to be assessed over time in collaboration with GISS developers.

The relation of the Cupid IDE to testing and validation of components is a critical one. There is an ongoing effort to introduce test driven development into the GISS model led by collaborator Clune, of the NASA Software Systems Support Office, using the Eclipse plugin for the Hudson software build environment. We will explore how Cupid can be integrated with the Hudson package to support improved testing of components.

In the following sections, we outline the primary objectives of our work, explain more about our motivation, methodology, and expectations, and describe the practical aspects of implementation and management.

\section*{1.1 Objectives and Expected Significance}

\subsection*{1.1.1 Objectives}

The technical objectives of the proposed work are to:

\begin{itemize}
  \item [OBJ1.] Develop the basic functional elements of the Cupid Integrated Development Environment, a software development and training tool for creating, configuring, and executing ESMF-based modeling applications.
  \item [OBJ2.] Introduce standard ESMF infrastructure for coupling and utilities into the GISS Model E code at the level of major physical domains (atmosphere, ocean, ice, etc.), with the intent to improve aspects of parallelization and performance, to improve modularity and thus controlled testing, to allow for the easier introduction of new components, especially ocean and ice models, and to automate code documentation through ESMF metadata handling utilities.
\end{itemize}
OBJ3. Create a set of demonstrations of running and using Cupid, spanning the execution of a straightforward code snippet to an end-to-end example showing how Model E can be modified, configured, augmented and then run.

The objectives align with these specific efforts of interest appearing in the ROSES solicitation, listed below in italics:

ROSES 2011 A40 Section 1.2.2.2: the components to support the use of Earth System Modeling Framework and other tools such as compiler, debugger, refactoring tool, and source code version control systems used for developing parallel codes on a supercomputer are not available. Funding will be provided to develop an open source IDE which will be available especially to the GEOS-5 and NU-WRF Earth system modeling community.

We propose to develop an IDE as described, for use within the NASA community and extensible to a range of ESMF-based models. GISS Model E was chosen as a starting point because it is relatively simple, and because its limited parallelism and lack of entrenched infrastructure suggest that its developers are more receptive to change than other more technically advanced codes.

ROSES 2011 A40 Section 1.2.2.3: Documentation is hard. Software engineering is a different discipline than most Earth system scientists have. Funding may be provided in the development of tools and environment to support the life cycle management of the model and data assimilation codes.

The proposing team will explore the incorporation of automated documentation into Model E as ESMF is introduced. Using the ESMF Attributes class, much of the exhaustive, hierarchical model metadata required for CMIP5 can be embedded in situ into a modeling code, and output as a standard XML schema (the “Common Information Model” or CIM). This schema can be ingested for archival purposes along with model data and for display by tools developed by the Curator and METAFOR/IS-ENES projects, in collaboration with the Earth System Grid Federation. It is a way for deeply linking provenance to the model software and sustaining it with the code over its lifecycle.

ROSES 2011 A40 Section 1.2.3: In the past couple of years, it has become increasingly apparent that many of the new environmental scientists entering the NASA workforce require additional training in parallel computing and software engineering. Funding will be available to develop an experimental summer school and internship program which augments the training at many Earth and environmental science programs with parallel computing, computational science, and software engineering.

The current proposal does not address this aspect of the solicitation directly. However, it will create a tool that teachers can utilize to train students about climate model development and use.

1.1.2 Expected Significance

- This project will demonstrate the viability of an IDE environment for development of Earth science modeling applications. This is an important step in making complex modeling more appealing and accessible to a new generation of engineers and scientists.
- An IDE is an entry point for Earth system model developers to harness the range of software tools being developed by the very active Eclipse open source community. Tools that will be explored immediately include the Parallel Tools Platform, Photran, and Hudson, for continuous testing and integration.
- The proposed work will improve the software structure of the GISS model, allow routine tasks to be automated, and make it a viable teaching tool.
The Earth System Modeling Framework

The Earth System Modeling Framework project was initiated by NASA in 2002. Over the last decade, it has become a widely used tool in the Earth Sciences, with more than 4000 downloads. It includes high-performance software for representing and coupling model components, and a set of utilities for common modeling functions. In addition, it includes software for parallel generation of interpolation weights that can be run as a separate application. This software requires only that the user provide grid files in a standard format.

The ESMF tools are unique in that they can represent logically rectangular grids, unstructured meshes, and observational data streams, and transform data among them by translating them each to an underlying unstructured representation. Their parallel implementation allows them to handle very large grids, and extensive regression testing means that they can operate on virtually all current high-performance computing platforms. This combination of capabilities is important because newer atmospheric model grids are often unstructured (e.g., HOMME, MPAS), high resolution, and may require coupling to more traditional logically rectangular ocean grids, such as the POP displaced pole grid. The ESMF grid transformation functions have proven critical for new research in climate codes such as CESM.

Models that have implemented ESMF interfaces include CESM, the Weather Research and Forecast model (WRF), NASA’s GEOS-5 atmospheric general circulation model, NOAA's National Environmental Modeling System (NEMS), Naval Research Laboratory's Coupled Ocean-Atmosphere Mesoscale Prediction System (C0AMPS) and the GFDL Modular Ocean Model (MOM4). ESMF is also used for regridding within TIMEGCM, a space weather code at NCAR, and within watershed and storm surge models developed by the U.S. Army. In general, ESMF componentization has been implemented at the level of major physical domains, where simulated interactions require inter-component data communications (e.g., atmosphere, ocean), and it has been implemented as wrappers that minimally modify existing user code. This implementation is compatible with codes such as WRF that impose their own interoperability conventions at the more fine-grained level of individual physics parameterizations. ESMF is supported and governed by a multi-agency consortium and is maintained as a production system. It is regression tested nightly on more than 30 platforms, is distributed with exhaustive documentation, and has dedicated support resources.

In ESMF, the user’s scientific code is “sandwiched” between two layers (see Figure 1). The layer above ensures a properly encapsulated model while the layer below provides a set of commonly used abstractions shared across all components in an application. The upper superstructure layer serves as a “unifying context within which user components are interconnected”. This layer provides an overall organizing structure for numerical model codes. Adopting the superstructure layer entails organizing model code into distinct gridded and coupler components, wrapping data to conform to certain standard interfaces (SetServices, Initialize, Run, and Finalize subroutines) and packing data that will be transferred among models into import and export units. The lower infrastructure layer is a set of both scientific and technical services commonly needed for ESMs such as calendar management, domain decomposition, logging, and error handling.
GISS Model E

Model E is an atmospheric general circulation model which is currently in its third major incarnation. In addition to atmospheric processes, it includes representation of land surface (snow, vegetation), land ice, lakes, and sea and lake ice. The model may be coupled to a dynamic ocean or may use simplified versions of ocean physics. The development philosophy has been to improve the physics of each model component, and to allow as great a degree of flexibility as possible in modeling configurations. The model supports standard resolutions used in the climate modeling community, as well as a $8^\circ \times 10^\circ$ mode useful for testing and teaching.

There has been a concerted effort to modernize the Model E code over the last decade. Fortran 90 constructs are favored and the code is increasingly modular. However, from a software engineering perspective it is not as sophisticated as models such as the Community Earth System Model (CESM) or the GFDL climate modeling suite, which support domain decomposition using a distributed memory programming model and MPI, or mixed modes of MPI and OpenMP. Model E has been parallelized with OpenMP. Although work has been done to parallelize the model using MPI, this work is not complete or functional.

The model can be downloaded from the web. The model is used for research at GISS and experiments run with the model have been submitted for Intergovernmental Panel on Climate Change assessments.

1.2 Technical Approach and Methodology

The technical approach will follow ESMF architectural standards and development processes that have evolved over its lifetime. These are described in the *ESMF Developer’s Guide* [11]. The proposing team will explore new strategies for collaboration and new technical approaches as appropriate. The following sections describe the technical approach relevant to each of the objectives.

OBJ1. Develop the basic functional elements of the Cupid IDE

IDEs have the ability to automate or assist users in performing routine development tasks. For example, the Eclipse IDE has been widely successful in supporting routine refactorings of Java programs, such as automatically inserting accessor and modifier methods for class variables. Users of Earth System Models (ESMs) also perform routine development tasks on the model, although to our knowledge, no IDE-like tools exist that help with these tasks. Examples...
of development tasks include (1) setting up a trace gas scenario, (2) adding a new parameter to the model, (3) adding a new diagnostic field to the model’s output, (4) adding a new physics parameterization, and (5) adding a new version of a component, such as a new ocean model. When a modification has been applied, the user initiates a workflow—which may be performed manually or automatically—in order to build, deploy, and execute the model, and analyze the resulting output.

As part of the Earth System Curator project, the collaborators on this proposal have already created an initial version of Cupid based on the Eclipse IDE. The current implementation lacks the basic infrastructure to meet the scientific requirements identified above. Therefore, the first objective is to develop and/or integrate the following functional elements into Cupid, using existing Eclipse plugins when possible.

- **Editor for the Fortran language.** The vast majority of existing ESMs are written in Fortran, including Model E. The Photran\textsuperscript{12} plugin for Eclipse is a maturing technology providing syntax highlighting, code outline views, content assist, and language-based searching capabilities. We will integrate the Photran editor into Cupid.

- **Code transformation engine.** Cupid will provide user assistance in performing routine development tasks that require changing a model’s source code. These will be implemented as a set of domain-specific code transformations. Even complex modifications requiring comprehensive changes to multiple source files can be decomposed into a set of simple code transformations such as adding a variable, calling a subroutine, or importing a module. Photran includes an existing abstract syntax tree rewriting engine that can be used to implement the generic Fortran transformations and refactorings. Having a robust library of generic code transformations will reduce the development burden required to implement domain-specific transformations for Model E or other ESMs. The class diagram in Figure 2 indicates how domain-specific transformations such as adding a new lat-lon diagnostic depend on generic Fortran code transformations. The set of supported code modifications on Model E is described in detail in Objective 3.

- **Remote system file management.** ESM source code created or modified within Cupid must eventually be transferred to the target compute cluster for compilation and execution. Cupid will utilize a third-party plugin such as the Remote System Explorer\textsuperscript{13} to manage secure copy and retrieval of files located on remote file systems.

- **Workflow engine.** Most ESMs rely on a set of text-based configuration files, makefiles, and shell scripts to direct model build and execution. The series of workflow tasks and their inter-dependences are best managed by a workflow engine specifically designed for this purpose. We will integrate Cupid with the existing NASA MAPME Workflow Engine. The integration tasks are described in detail in the following section.
Integration of the NASA MAMPE Workflow Engine

For the build, deployment, and monitoring phases of the modeling process, Cupid will interface with the NASA MAP Modeling Environment Workflow Engine (NWE). NWE is a tool designed to manage modeling experiment workflows of NASA models. NWE workflows are configured via a simple GUI and the workflow engine manages task execution automatically while providing feedback to the user about the status of the workflow. Leveraging such a tool is essential for training environments in which users have limited knowledge of executing high-performance applications on supercomputing clusters.

Architecturally, NWE is based on two major components: the NASA Experiment Designer (NED) client and server, and the Light-Weight Workflow Engine (LWWE) server component. The client NED allows users to create workflow configurations in a user-friendly environment. The NED server receives configurations from clients, prepares the workflow working directory, and executes the workflow using LWWE. There are two major inputs to NED: a workflow configuration file and a workflow architecture file. The configuration file contains all of the options that the workflow developer would like the user to be able to change while setting up the workflow. The workflow architecture file contains information about all the tasks required to execute the workflow and inter-dependencies among the tasks. This information is also used to display structured monitoring data to the user as the workflow executes.

Both NED and LWWE are portable Java-based components. The following tasks are required to integrate Cupid with NWE:

- Design and develop a new NED client as an Eclipse plugin. The plugin will interact with NED servers via the existing XML message-based interface. The plugin will send messages to the NED server to begin a workflow execution and will wait for response messages to update the user on the status of the workflow. Because the current NED client implementation is Java-based, it may be possible to reuse some of the GUI components in the Eclipse environment.

- Implement a mechanism for exposing source code generated and configured by Cupid to the NED server. The NED server expects all files required for a workflow, including source code, to be accessible via source control or on the target platform’s file system.
Therefore, before executing the workflow (or as an initial workflow task), Cupid-generated source code must be transferred to the target platform either via secure file transfer or check-in to source control.

- Create a set of canned workflow architectures for the supported training models and configurations, including a basic radiative-convective model and Model E. This ensures that Cupid will have a set of pre-validated workflows “out of the box” for new users.
- Devise a basic validation engine that examines monitoring data returned from LWWE and model output and notifies the user of a successful run. In the event that a run fails, the user will be notified of the source of the problem and provided some hints for how to make corrections.

**Current Status of Cupid**

The current version of Cupid generates ESMF coupler and driver components based on a coupling configuration provided by the user. The generated code 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. A graphical user interface provided by the Eclipse Modeling Framework allows the user to input the configuration graphically.

Figure 3 shows a screenshot of the existing Cupid graphical editor. This interface enables users to create a specification of existing ESMF components and to define the coupling configuration between them by manipulating domain-level objects in a hierarchical structure. In this example, there are two Gridded Components (user_model1 and user_model2) and a Coupler Component (gencpl). The distributed data structures within the components are also defined in this interface (e.g., dataArray and distGrid_user1) as well as the mapping of coupling fields between components. The graphical representation is converted into an XML format suitable for storage and input into the code generator. The prototype version of Cupid validates the feasibility of generating coupling code for ESMF. However, significant work is left to be done before the software is mature enough for real scientific applications.
OBJ2. Introduce ESMF infrastructure and utilities into Model E

Integration of ESMF into Model E will begin with introduction of ESMF component interfaces on the main model components: atmosphere, ocean, and ice modules. These will be implemented as wrappers, so that the system can operate without ESMF for simple configurations. The NUOPC Layer will be demonstrated to GISS developers to determine if they would like to explore its use in their production code.

To the extent possible, development work will be aligned with advancing the scientific goals of GISS modelers. If the Model E group has interest in better defining the interface to atmospheric dynamical cores, this additional level of componentization will be pursued. Early experimentation has already taken place in this area at GISS. This effort may also be pursued on an experimental track in order to demonstrate the use of Cupid.

An experimental module will be chosen to demonstrate the implementation of embedded metadata handling. Atmospheric radiation is a good starting point because there is a limited amount of CMIP5 metadata associated with short and long wave processes, and the modules are usually well defined. The approach will be discussed with the Model E team to determine whether and how the work could extend to other parts of the code.

Other utilities, such as calendar management and grid remapping, may offer features that Model E does not yet have and will be pursued if there is interest. A major consideration in determining the level of integration is whether ESMF will be required to build Model E or not. A goal of the proposed work will be to ensure that the top-level component interfaces are routinely tested and used for coupled configurations.

OBJ3. End-to-end workflow within the Cupid IDE

The Cupid IDE sits in between highly-flexible, expert-friendly XML-based workflow systems such as GFDL’s FMS Runtime Environment and point-and-click, novice-friendly systems such as the Perdue CCSM portal. The goal of these systems is to increase productivity by automating parts of the workflow. We will build tools within Cupid to address several parts of the scientific workflow, including assisting the user in making routine code modifications, providing the user with a high-level outline view of the component architecture of ESMF-based models, and providing continuous integration compliance checking.

Within the ESM domain, conventions have emerged that constrain the model development process in terms of coding practices, ways of using third-party libraries, naming rules for variables and files, file formats, example code fragments for routine modifications, and documentation standards. These conventions range in formality and scope. Simple models such as a radiative-convective simulation or a test code snippet that will eventually evolve into a physics parameterization may not adhere to any particular conventions outside the programming language itself. As a model increases in sophistication and size, model-specific conventions emerge, such as an agreed upon way of accessing all parameters from a centralized database or locating all of the diagnostic field metadata in the same file. Framework-specific conventions, such as those provided by ESMF, are constraints or best practices that arise as a result of using a software framework. These conventions are general enough to apply to an entire domain. An example of a framework-specific convention derived from ESMF is the use of separate gridded components for each geophysical domain and coupler components as mediators between geophysical components. Finally, community-specific conventions, such as ESMF’s usability layers, are a refinement of framework-specific conventions for a particular set of users within the domain who desire to promote code reuse and increase interoperability of components.
In the following section we give specific examples of how constraints implied by the sets of conventions mentioned above enable Cupid to assist users in various stages of the end-to-end workflow.

**Raw code snippets**

Raw, unstructured code snippits provide limited semantics to Cupid. Cupid will handle this code in one of two ways. First, just as within any general purpose IDE, the user can provide a custom workflow for how to configure, compile, and execute the code snippit and interpret the output herself. Secondly, Cupid will provide the option to wrap unstructured code inside an ESMF component. In this case, the user will be required to ensure that the component’s interfaces are correctly implemented.

**Model-specific conventions enable automation of routine development tasks**

Model-specific conventions enable automation of routine development tasks within a particular model. We propose to build features within Cupid to support the following routine development tasks on Model E. The Model E How-To document\(^1\) and Developer’s Guide\(^2\) describe how to perform these tasks manually.

**Adding a new parameter to Model E**

Model E has recently adopted a common mechanism for setting and accessing the model’s parameters. All parameters are stored in a parameter database and they can be accessed via calls to the parameter database library. This improves readability of the code and simplifies model restarts. As outlined in the How-To document, the basic steps for adding a parameter are:

1. Decide which module should own the parameter. This is likely one of the physics modules.
2. Add a Fortran variable for the parameter in the owning module, set a default value, and include a custom annotation ‘@dbparam’ that explains the parameter.
3. Add a call to `sync_param()` in the initialization portion of the module that owns the parameter.
4. Add the parameter to the rundeck between the `&&PARAMETERS` and `&&END_PARAMETERS` identifiers.
5. Add calls to `get_param()` and `set_param()` in order to read and write parameters in the parameter database.

Although the steps for adding a parameter are straightforward, the user is required to modify at a minimum two files (the owning module and the rundeck) and potentially others (wherever the parameter is used) and could accidentally overlook some requirements such as setting the default value or adding the `@dbparam` annotation.

Cupid includes a feature that assists the user in the steps above. When the user selects “Add Model Parameter” from the menu, the user will be asked to identify the owning module, which

---

\(^1\) [http://www.giss.nasa.gov/tools/modelE/HOWTO.html](http://www.giss.nasa.gov/tools/modelE/HOWTO.html)

\(^2\) [http://www.giss.nasa.gov/staff/gschmidt/modelE_develop.html](http://www.giss.nasa.gov/staff/gschmidt/modelE_develop.html)
will be defaulted to the file that currently has the focus. Then, the user will be asked for the name of the parameter, the default value, and a brief description. Cupid will then automatically add a new local variable to the owning module, set the default value and \texttt{@dbparam} annotation, insert a call to \texttt{sync_param()} for the new parameter, and add the parameter to the rundeck.

Adding a new lat-lon diagnostic output to Model E

The following steps are required to add a new lat-lon diagnostic output to Model E.

1. Declare the new diagnostic by adding a new integer variable \texttt{ij\_mydiag} to \texttt{DIAG\_COM.f}. This variable holds the unique index for the diagnostic.
2. Add metadata about the new diagnostic such as the name, units, and scale factor into the file \texttt{DEFACC.f}.
3. To include the diagnostic in the ASCII output, add the diagnostic’s unique index to the array \texttt{iord} in \texttt{DIAG\_PRT.f}.
4. Add a new subroutine that calculates the value of the diagnostic.

The “Add New Lat-Lon Diagnostic” feature in Cupid will automate this process by presenting the user with a form for gathering information about the new diagnostic including its name, units, scaling factor, weighting, whether it should appear in the ASCII output, and all other relevant metadata. Then, the tool will make modifications to the appropriate files identified above, including the adding the new diagnostics index variable in \texttt{DIAG\_COM.f}, adding the metadata to \texttt{DEFACC.f}, optionally adding the diagnostic’s index to \texttt{DIAG\_PRT.f} if the user elected to include the diagnostic in the ASCII output, and creating a new file \texttt{DIAG\_MYDIAG.f} with a subroutine template where the user provides code for calculating the diagnostic.

Adding a new physics parameterization to Model E

Cupid will support adding a new physics parameterization to Model E. Fortunately, the process can be simplified because a set of coding conventions exist that standardize the way physics modules are interfaced. To add the new physics package, the user is required to implement three files: the local physics module itself (e.g., \texttt{MYPHYSICS.f}), which is self-contained and does not reference the main model variables directly, the physics interface (e.g., \texttt{MYPHYSICS\_COM.f}), which owns the model variables related to the local physics, and the physics driver (e.g., \texttt{MYPHYSICS\_DRV.f}), which provides initialization routines, calls the local physics calculation routine, and finally updates the main model variables with new fluxes, sources, or sinks.

Cupid will include a “New Physics Parameterization” feature that assists the user in the steps above. This feature will be implemented as a “wizard” that walks the user through the steps described above providing templates for all three required files. Cupid will reduce the likelihood of coding errors by allowing the user to select from a list of main model variables instead of requiring the user to remember them or look them up. Additionally, Cupid will maintain metadata about the new physics package as a whole and the implied relationship between the three files \texttt{MYPHYSICS.f}, \texttt{MYPHYSICS\_COM.f}, and \texttt{MYPHYSICS\_DRV.f}. This metadata will be used to check compliance with the coding standards, such as ensuring that the driver only calls routines inside the local physics implementation and verifying that main model variables are not accessed directly inside the local physics subroutines.
Framework-specific conventions enable manipulation of ESMF components

The superstructure layer of ESMF provides an architecture in which components within an ESM may be exchanged with new components that share the same interface. This capability is important for testing the effects of different numerical schemes or different formulations of physics. As an example, the graduate-level “Art of Climate Modeling” course taught at the University of Michigan by Professor Christiane Jablonowski requires students to explore one component of a general circulation model: the atmospheric dynamical core (sometimes shortened to “dycore”), which resolves the fluid flow equations on the discretized sphere. The students are required to test the effects of applying different idealized initial conditions to one of several dynamical core components available in CESM. By exchanging one dycore for another, the effects of different numerical methods and advection schemes become clear.

We envision a scenario in which students perform a similar set of experiments within Cupid. The scenario is based on having a version of the NASA GISS Model E componentized with ESMF. On startup, the IDE shows a tree view of the set of components within Model E. A default configuration using all components is shown to the user. For training purposes, the user wishes to focus on the dynamical part of Model E, so she selects the “simplified physics” category of the tree. She right clicks on the dynamical core component and selects the “Configure...” option. She then selects a number of basic parameters such as the start date, stop date, and location of the initial conditions file. The user right clicks on the dynamical core component again and selects “Generate Driver.” A new ESMF component source file is automatically generated for driving the dynamical core. The user is now ready to compile and execute the model. A Model E workflow has already been created using the NASA MAPME Workflow Tool and the user is able to access the workflow directly from Cupid’s menu. The user clicks “Execute Workflow.” Messages are sent to the NASA Workflow Tool server on the compute cluster. Meanwhile, the IDE shows the user a new screen where she can monitor the status of the workflow and examine any log files output from the model. When the workflow is completed, the IDE displays the location of the NetCDF files generated by the model.

Community-specific conventions enable continuous, automated compliance checking

In addition to the basic architectural constraints of the ESMF framework, some communities have defined and implemented an additional layer of architectural constraints and coding conventions on top of ESMF. For example, the Navy and NOAA have jointly adopted a set of conventions codified in the ESMF NUOPC usability layer. To ensure that implementations are compliant, an automated compliance checker has been implemented within ESMF. Information is provided to the user in a semi-structured format in log files generated by the model.

Cupid will promote the continuous integration development practice by incorporating the ESMF compliance checking tool with the model build and testing process. We will explore the Hudson Continuous Integration server\(^\text{14}\) to determine if it provides adequate infrastructure for automating the ESMF compliance checking. Cupid will provide frequent and specific feedback to the user about any compliance violations. Output from the ESMF compliance checker localizes violations to a specific line of code in some cases. Cupid will parse this information and direct the user to places in the code that require attention.

1.2.1 Testing

There are a variety of different types of software being developed as part of the proposed work. The Cupid software will have a graphical interface with many options and usage
sequences. This is a difficult type of software to test thoroughly. Further, whether users perceive the software is working properly or not is likely to depend on whether the interface is designed intuitively. In early stages of development, it can also be difficult to set user expectations about the limits of the software. This is especially challenging in a system that is designed to steer the user without a lot of text instruction.

Practices that are designed to prevent user disappointment are: 1) the creation of use cases in collaboration with prospective users; 2) careful discussion and descriptions of individual capabilities, with mockups where possible; 3) regular demonstrations, covering demonstration of every new capability before work is determined complete; 4) beta test periods in which the status of the software is clearly indicated; 5) to the extent possible, in situ help in the user interface; and finally 6) careful monitoring and tracking of user reports.

Integration of ESMF software into Model E presents a different set of challenges. We plan to identify and document a set of minimum tests that must be completed before submitting a set of changes to the Model E development team for review and incorporation into their production version. Deliveries of a set of software changes will be accompanied by a report that includes all relevant software versions, computing platform, a report of which tests were executed (additional tests besides the minimum may be warranted), the results of those tests, and any notes. The Model E team will then be encouraged to run their standard acceptance tests. This strategy is being followed successfully for integration of ESMF into other models.

The ESMF distribution itself includes a test suite with 4000+ unit tests, system tests, and examples. The test suite is regression tested nightly on 30+ platform and compiler combinations. An automated script runs weekly to ensure that there is full test coverage of all public interfaces. Further, a suite of “use test cases” in which ESMF methods are tested on realistic problem sizes is run regularly.

1.3 Perceived Impact to State of Knowledge

The work proposed builds on and extends software capabilities and partnerships that were successfully established over the last decade under the leadership of the PIs with support from NASA. During that time the ESMF was created and adopted by leading U.S. weather and climate models, and in the process standardized to an unprecedented degree the way in which U.S. model components are constructed.

This proposed work will provide new insights into how advanced development environments can serve as training tools for novice Earth system modelers and how such tools can lead to increased scientific productivity for intermediate and advanced modelers. To our knowledge, no modeling centers are using IDE-based approaches comprehensively—most have implemented customized build systems based on text-based configuration files and shell scripts. Unfortunately, this means that much effort has been spent “re-inventing the wheel” since the abstract requirements for model configuration and execution are universal. To this point, the idiosyncrasies of specific model architectures and coding practices have necessitated the highly customized workflow solutions. By designing the Cupid code modification and workflow features as generically as possible, we hope to uncover some of the common conceptual requirements for ESM workflows and provide some implementation components within the Eclipse environment that can be extended to other models outside of GISS.

The identification of the role of model-specific, framework-specific, and community-specific conventions in aiding the development of ESMs is an important contribution to the state of knowledge. Elements within each of these types of conventions will be codified into Cupid and we suspect that these and similar conventions will play an increasingly important role in the Earth system modeling community. The advantages of code reuse, component interoperability,
and improved maintainability of models will hopefully influence modeling centers to adopt more rigorous conventions and codify them into tools like Cupid that help improve scientific productivity.

1.4 Work Plan

We will use the established ESMF development process, in which design and implementation reviews are conducted using distributed development approaches (conference calls, desktop sharing software, mailing lists, etc.). The technical mailing list for ESMF, on which information about reviews is broadcast, now numbers about 650 members and has broad participation. These reviews will be critical to identifying additional customers and obtaining feedback. Using an approach modeled on the established ESMF support process, which has handled thousands of user requests, we will work with early users to evolve the Cupid software to the point at which it can be distributed and supported more broadly. The eventual goal is to demonstrate value and petition for the ESMF-relevant portion of Cupid to be distributed through the ESMF core development team and maintained through its multi-agency sponsorship.

1.4.1 Key Milestones

The anticipated start date for the proposed work is FY 2013-1Q.
- Completed integration of functional elements into Cupid, including the Fortran language editor, code transformation engine, remote file manager – FY2013-3Q
- Completed integration of NASA MAPME Workflow Engine including validation with basic radiative-convective model – FY2013-4Q
- Initial componentization of Model E with ESMF infrastructure – FY2014-1Q
- Model-specific transformations for Model E implemented – FY2014-1Q
- Framework-specific (ESMF) component outline implemented – FY2014-2Q
- Compliance checker implemented – FY2014-3Q
- Demonstration and validation of fully ESMF-ized Model E in Cupid workflow – FY2014-4Q

1.4.2 Management Structure

Cecelia DeLuca of the University of Colorado is the PI of the proposed investigation. She is responsible for the execution of the proposed activity, the quality of products delivered, and the proper use of all awarded funds. She is also responsible for all technical, management, and budget issues and is the final authority for this project. The Co-Is report to and take direction from the PI and will provide the information needed to ensure that she can effectively manage the project.

Spencer Rugaber and Leo Mark are Co-Is from the Georgia Institute of Technology, and Thomas Clune and Gavin Schmidt are collaborators from the NASA Goddard Space Flight Center and NASA Goddard Institute for Space Studies (GISS). The Co-Is and collaborators will coordinate activities within their organizations.

The PI and Co-Is will operate the project in the context of the existing ESMF multi-agency management structure. This is comprehensively described in the ESMF Project Plan, recently updated for 2010-2015.
1.4.3 Contributions of Principal Investigator and Key Personnel

PI Cecelia DeLuca is the manager of the ESMF development group and a point of contact for NUOPC technical activities. She will oversee the proposed effort and direct development within the ESMF collaboration, will work to ensure that Cupid becomes a production quality product, and will promote Cupid within the ESMF community and other venues.

Co-Is Leo Mark and Spencer Rugaber are professors of Computer Science at the Georgia Institute of Technology. They will direct students and other staff who are involved in the development of Cupid, and will help the collaboration to understand the current trends and issues that relate to this work.

1.4.4 Collaborators and Consultants

Collaborator Thomas Clune is Chief of the Software Systems Support Office (SSSO). He and his staff have contributed to software engineering and development of Model E, and will help to coordinate the Cupid work with continued development efforts within NASA. Dr. Clune will help to inform how the Cupid development group can integrate and interact with model testing.

Collaborator Gavin Schmidt from the Goddard Institute for Space Studies is a climate scientist and one of lead developers of Model E. He will serve as a point of contact for Model E and will help to coordinate activities with the development team.

1.4.5 Risk Management

The major risks associated with this development are that the Cupid capabilities desired are not delivered, and that the interaction with GISS does not result in them using either the ESMF or Cupid software. Software development processes for the project are designed to mitigate these risks.

The Cupid capabilities desired are not delivered. This could occur through incorrect prioritization of development tasks, misunderstanding of task requirements, poor estimation of resources and time required to perform tasks, faulty design or implementation of software, or insufficient management oversight to ensure that tasks are completed. These risks are largely controllable. The strategies listed under Testing (Section 1.2.1) are designed to reduce them.

The Model E team does not incorporate ESMF or Cupid into routine processes. These risks may be due to technical difficulties in building or using ESMF or Cupid; resistance to code that is “not developed here” or code patterns that look unfamiliar; lack of resources; or lack of sufficient expertise. Good communication among teams, realistic expectations, and clear, written, mutually agreed on integration plans will help to minimize these risks, as will aligning software and infrastructure development with scientific goals to the extent possible.

1.5 Management of Source Code and Information

Cupid will be distributed under an open source license and maintain a development repository that is publicly viewable. Releases will be available for download from the web.

The Cupid team is committed to providing as much information about the project as feasible in an open, accessible way. To the extent possible, documents related to the project will be publicly viewable.
2 References and Citations

ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CAN</td>
<td>Cooperative Agreement Notice</td>
</tr>
<tr>
<td>CCSM</td>
<td>Community Climate System Model</td>
</tr>
<tr>
<td>CESM</td>
<td>Community Earth System Model</td>
</tr>
<tr>
<td>CIM</td>
<td>Common Information Model</td>
</tr>
<tr>
<td>CMIP5</td>
<td>Climate Model Intercomparison Project 5</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modeling Framework</td>
</tr>
<tr>
<td>ESM</td>
<td>Earth System Model</td>
</tr>
<tr>
<td>ESMF</td>
<td>Earth System Modeling Framework</td>
</tr>
<tr>
<td>FRE</td>
<td>Flexible Runtime Environment</td>
</tr>
<tr>
<td>GEOS-5</td>
<td>Global Earth Observing System 5</td>
</tr>
<tr>
<td>GFDL</td>
<td>NOAA Geophysical Fluid Dynamics Laboratory</td>
</tr>
<tr>
<td>GISS</td>
<td>Goddard Institute for Space Studies</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HPC</td>
<td>High Performance Computing</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IS-ENES</td>
<td>Infrastructure for the European Network for Earth System Modeling</td>
</tr>
<tr>
<td>LWWE</td>
<td>Light-Weight Workflow Engine</td>
</tr>
<tr>
<td>MAPL</td>
<td>Modeling Analysis and Prediction Layer</td>
</tr>
<tr>
<td>MAPME</td>
<td>MAP Modeling Environment</td>
</tr>
<tr>
<td>MOM4</td>
<td>Modular Ocean Model 4</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NCAR</td>
<td>National Center for Atmospheric Research</td>
</tr>
<tr>
<td>NED</td>
<td>NASA Experiment Designer</td>
</tr>
<tr>
<td>NOAA</td>
<td>National Oceanic and Atmospheric Administration</td>
</tr>
<tr>
<td>NSF</td>
<td>National Science Foundation</td>
</tr>
<tr>
<td>NUOPC</td>
<td>National Unified Operational Prediction Capability</td>
</tr>
<tr>
<td>NWE</td>
<td>NASA Workflow Engine</td>
</tr>
<tr>
<td>WRF</td>
<td>Weather Research and Forecasting Model</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
3 Biographical Sketches

Principal Investigator

CECELIA DeLUCA
NOAA Earth System Research Laboratory
325 Broadway
Boulder, CO 80305
cecelia.deluca@noaa.gov

Education
A.L.B. 1992  Liberal Arts/Social Sciences,
Harvard University, Cambridge, MA
M.S. 1994  General Engineering,
Boston University, Boston, MA
M.S. 1996  Meteorology,
Massachusetts Institute of Technology, Cambridge, MA

Positions
2009-Present  Group Head, NOAA Environmental Software Infrastructure and Interoperability
NOAA Cooperative Institute for Research in the Environmental Sciences
2002-Present  Manager, Earth System Modeling Framework Core Team
2006-2009  Section Head, Earth System Modeling Infrastructure
National Center for Atmospheric Research
2004-2006  Group Head, Earth System Modeling Framework
National Center for Atmospheric Research
1999-2004  Software Engineer /Technical Project Manager, ESMF
National Center for Atmospheric Research
1996-1999  Software Engineer
MIT Lincoln Laboratory

Selection of Peer Reviewed Publications
integrated Earth system models: A coupled atmosphere-ocean modeling system application,
Environmental Modeling & Software, in press.
2. Zhao, L., C. Song, C. Thompson, H. Zhang, M. Lakshminarayanan, C. DeLuca, S. Murphy, K. Saint,
D. Middleton, N. Wilhelmi, E. Nienhouse, and M. Burek (2011). Developing an integrated end-to-end
TeraGrid climate modeling environment , TeraGrid ’11, Salt Lake City, Utah, July 18-21, 2011.
Received award for best paper in the Science Gateways track.
Murphy (2008). Earth System Curator: Metadata Infrastructure for Climate Modeling. Earth
Science Informatics, Volume 1, Numbers 3-4, pp. 131-149.
International Journal of High Performance Computing Applications, Volume 19, Number 3, pp. 341-
350.
Co-Investigator

Spencer Rugaber  
Senior Research Scientist  
College of Computing, Georgia Institute of Technology  
Atlanta, GA 30332-0280  
(404) 894-8450

RELEVANT EXPERIENCE

Dr. Rugaber is a Senior Research Scientist in the College of Computing at the Georgia Institute of Technology. He was previously the Georgia Tech Principle Investigator of the NSF-funded Earth System Curator project. This effort was joint with the National Center for Atmospheric Research and the Geophysical Fluid Dynamics Laboratory. He has also served as PI or co-PI on numerous projects funded by NSF, DARPA, the Army Research Laboratory, the Institute for Education Sciences, and industrial organizations. During 2001-2002, he served as Program Director for the National Science Foundation program on Software Engineering and Programming Languages. He has published over seventy articles describing his research in software engineering, modeling, human-computer interaction and artificial intelligence.

EDUCATION

Ph. D., Computer Science, Yale University, 1978  
M.S., Applied Mathematics, Harvard University, 1971  
B.S., Engineering and Applied Science, Yale University, 1970

CURRENT POSITION

1988-present: Research Scientist, College of Computing, Georgia Institute of Technology

SELECTED REFEREED PUBLICATIONS

Co-Investigator

Leo Mark
Associate Professor
College of Computing, Georgia Institute of Technology
(404) 385-7560

RELEVANT EXPERIENCE
Dr. Mark is an Associate Professor in the College of Computing at the Georgia Tech. He is the Georgia Tech Co-Principal Investigator of the NSF-funded Earth System Curator project, a joint effort with NCAR and GFDL. He was a member of the Committee on Geophysical and Environmental Data, under the Board on Earth Sciences and Resources, National Research Council from 2002 to 2003, and in 2003 he gave the invited keynote address, "Government Data Centers - Meeting Increasing Demands," at the NASA EOS SWGD Data Access & Usability Workshop, in Greenbelt, Maryland. From 2003 to 2004, he served as a consultant for Accenture’s effort to win the National Archives and Records Administration’s (NARA) ERA Procurement. Dr. Mark has served as PI or Co-PI on numerous projects funded by NSF, DARPA, ARL, DMSO and NASA. He has published over seventy articles in databases, metadata management, and temporal databases.

EDUCATION:

Ph.D., Computer Science, Aarhus University, 1985
M.S., Mathematics, Aarhus University, 1980

PROFESSIONAL EXPERIENCE:

2005-2010: Director, International and Professional Programs, College of Computing, Georgia Tech
1992-present: Associate Professor, College of Computing, Georgia Tech
1986-1992: Assistant Professor, Computer Science Dept., Univ. of Maryland

SELECTED REFEREED PUBLICATIONS

Collaborator

Dr. Thomas L. Clune
Senior Computational Scientist
Software Systems Support Office
NASA Goddard Space Flight Center, Greenbelt, MD 20771
(301) 286-4635
thomas.l.clune@nasa.gov

Education:
1993 – Ph.D. in Physics, University of California at Berkeley
1990 – M.A. in Physics, University of California at Berkeley
1988 – B.S. in Physics, Massachusetts Institute of Technology

Special Experience:
Dr. Thomas Clune is the Chief of the Software Systems Support Office (SSSO) within the Earth Science Division at NASA Goddard Space Flight Center. His office is responsible for a variety of activities centered on supporting design, modernization, componentization, maintenance, and optimization of scientific software. Dr. Clune consults for several global Earth Science modeling teams including the Goddard Institute for Space Studies, Global Modeling and Assimilation Office (GMAO), and NASA Unified WRF (NU-WRF).

For the past decade Dr. Clune has worked closely with the developers of GISS ModelE to improve the capabilities of their climate model through the introduction of advanced software techniques. Early work with ModelE included the introduction of MPI and domain decomposition to enable parallelism on computing clusters. More recent work has focused on introducing unit testing and automated regression testing. Current activities include the application of GPU hardware accelerators, re-engineering infrastructure for robust support of tracers and diagnostics, and componentization of the primary physical subsystems.

Related and Selected Publications:


Collaborator

Gavin A. Schmidt
NASA/Goddard Institute for Space Studies, Code 611, 2880 Broadway, New York, NY 10025
Email: Gavin.A.Schmidt@nasa.gov Tel: (212) 678-5627

Education
BA (Hons), 1987, Oxford University, UK (Mathematics)
Ph.D., 1994, University College London, UK (Applied Mathematics)

Employment
2004 – present Physical Scientist, NASA Goddard Institute for Space Studies
2004 – present Adjunct Sr. Research Scientist, Center for Climate Systems Research, Columbia University
1996 – 1998 NOAA Postdoctoral Fellow in Climate and Global Change
1994 – 1996 Postdoctoral Fellow, McGill University, Montreal

Current Professional Activities
NCAR CSM Community Advisory Board: 2009 -
Member, CLIVAR/PAGES Intersection Panel 2004 - (co-chair 2007 - )
Associate Editor: J. Climate

Outreach
American Museum of Natural History, NY Academy of Sciences, College de France, RealClimate.Org,
CNN, ABC, Comedy Central, NY Times, Wash. Post, USA Today, New Scientist, and other media outlets.

Select Recent Publications


References and Citations


5. A popular example of this is the Eclipse Parallel Tools Platform (http://www.eclipse.org/ptp/).


