Contents

(I) Science, Technology and Management

1 Introduction

We propose to construct an Earth System Modeling Framework (ESMF) to provide high-performance, extensible and interoperable codes for weather prediction, climate simulations, and data assimilation. The multi-institutional and cross-disciplinary interactions among the broad community involved in the proposed work, together with the greater ease in composing robust and exchangeable components that the framework will provide, will result in a new generation of diverse and performance portable earth science applications.

The ESMF will constitute a concerted response to application development in a volatile high performance computing environment. For the last five years, the earth science modeling community has been involved in a transition from low-parallelism, shared-memory, vector computers to massively-parallel, distributed-memory, cache-based computers. This transition has been difficult and expensive, and is, in large part, not yet completed at most centers. This experience has shown that the community is ill-prepared to deal with major changes in high-performance computing and communications technologies. In particular, it has highlighted major weaknesses in the design of large earth science computational codes and in the community's ability to organize itself to deal with these weaknesses in a concerted way. The ESMF will focus resources on creating a powerful, flexible infrastructure that can assist the community in working effectively in the current environment and adapting more easily to future shifts in the HPC marketplace.

The framework will consist of component coupling services, a set of data constructs that support operations on a variety of data grids and decompositions, and a portable, optimized set of low-level utilities. General purpose utility routines will include a standard interface to a variety of I/O packages, performance profiling, time management, and error handling. The data constructs and low-level utilities will be used by the coupling services and may also be used separately to compose scientific applications. Thus our work will promote software reusability as well as interoperability.

The framework will be designed in an object-oriented, layered manner to isolate machine dependencies, and offer an application programming interface natural for the earth sciences. Several of the participants in this proposal have previously developed limited domain frameworks and toolkits for high-performance parallel computing (FMS, GEMS, SCRIP, PETSc, STAPL). We will leverage the work in these packages to accelerate the development of ESMF. In some cases, this will mean building directly on existing software; in others it will mean using existing packages as a basis for rapidly prototyping ideas.

The "critical mass" of institutions participating in this proposal originates in the Common Modeling Infrastructure Working Group (CMIWG). The CMIWG has attracted broad participation from major weather and climate modeling groups in Federal research laboratories, universities and operational centers. For the past two years, the group has discussed ways of enhancing collaborative earth system model development through a common infrastructure. At a CMIWG workshop in February 2000, participants committed to developing a coordinated response to the NASA HPCC-ESS CAN, seeing a tremendous opportunity to begin work on what will hopefully be the key element of a coordinated national earth system modeling strategy. Since then, the group has met regularly and has established a set of requirements and a preliminary design for the framework. The group has also entrained computer scientists with relevant experience and a strong interest in this application area.

In anticipation of this opportunity, we began the planning and design of an ESMF that would serve the varied needs of the institutions represented in this proposal. We have chosen to present our plan in three tightly interconnected proposals. This one (Part 1) deals with the core development work of the framework. Parts 2 and 3 deal with the specifics of modeling and data assimilation applications, respectively. Although the later parts focus on the applications—how they will interact with the framework and what contraints they place on its design—all three proposals have a single goal: the design and implementation of the framework *and* its establishment as the standard in earth science applications. Separating the description of our task into these three parts is somewhat at odds with the institutional separation envisioned in the Announcement. Nevertheless we adopted it because we felt it better reflected the advanced state of coordination we have already achieved and allowed for a clearer and more coherent presentation of our plan.

2 Scientific Objectives and Rationale

2.1 Goals of the ESMF

[Think this section needs a stronger tie to specific scientific applications/results.]

It is important that we present a rigorous case for the development of a community framework. A framework will involve political and administrative overheads and will, inescapably, require some compromises in performance and flexibiltiy, from all who comform to it. If the benefits do not outweight these costs, the project is, rightly, doomed to fail.

The objective of the ESMF is to provide the following benefits:

- **Facilitate the exchange of scientific codes (interoperability)** so that researchers may more easily take advantage of the wealth of resources that are available in the US in smaller-scale, process modeling and to more easily share experience among diverse large-scale modeling efforts. This may be summarized as doing a better job of taking advantage of *useful* diversity.

  Currently we can exchange codes at a very low level—the single-column physics, following the so-called plug-compatible rules. A number of measures are being taken by the Common Modeling Infrastructure Working Group (CMIWG) to facilitate these exchanges. We certainly do not need a more sophisticated framework for this.

  An ESMF would be aimed at higher-level, more "integrated" computational components. It is probably easier to port an entire atmospheric GCM—to take an example of extreme integration—to a coupled climate system, than to exchange the convection parameterization within a GCM. The latter requires much more "physical compatibility"—how does the convection couple to the turbulence, the cloud scheme, etc. At a higher level of integration, processes that are strongly coupled are placed together and, hopefully, treated in physically consistent ways by the designer. In coupling more integrated components, physical considerations are lessened, while logistic considerations are increased, since data structures are more complicated. It is the coupling and exchange of these "integrated components" that should the province of the ESMF.

- **Promote the reuse of standard, non-scientific software**, the development of which now accounts for a substantial fraction of the software development budgets of large groups.

  Any center developing or maintaing a large system for NWP, climate or seasonal prediction, data assimilation, or basic research will have to solve very similar software engineering and routine computational problems. Diversity in this area is not nearly as useful as in scientific aspects of these systems. In these tasks, we are much more likely to be "reinventing the wheel" than maintaining useful alternatives.

- **Focus community resources** to deal with architectural changes and the lack of quality commodity middleware. The non-scientific parts of the codes that would be dealt with in a common framework are also the most sensitive to architectural changes and "middleware" quality. The great fluidity of computing tecnology and the stesses of the market demand that vendors concentrate on commodity products to the detriment of high-end users. This will continue for the foreseeable future. When industry is unable or unwilling to provide the kind of support we require, we will need to be in the best possible position to "fend for ourselves." A framework is one means of organizing community collaboration to deal with these problems.

- **Present the computer industry with a unified, well defined and well documented task** for them to address in their software design. As noted already, the scientific community's influence with the industry is much diminished, but it will be even smaller if it is exercised separately by five or six centers.

  If the major centers continue to maintain different systems, vendors will have to divide their already meager benchmarking and support capabilities. By providing them a more unified problem to address, we not only pool these resources, but we are more likely to attract vendor's interest in the Earth Science market.

- **Share the overhead costs** of the housekeeping aspects of software development: documentation and configuration management. These are the efforts that are most easily neglected when corners have to be cut.

- **Provide institutional continuity** to model development efforts. Most US modeling efforts are necessarily tied to only a few individuals, and centers are hard-pressed to maintain continuity that transcends them. The competitive job market we are in will result in shorter tenure for programmers. At the same time, the increasing complexity of both our systems and the technology will produce more reliance on software specialists—and less on scientists—to maintain these aspects of the systems. Both of these factors will contribute to a more unstable workforce and much greater difficulty in maintaining "institutional continuity." A framework can helps us do this by having a much larger institution to support it—the whole community. Also by having codes depend on a common, well-known framework, it will be easier to find and train new people to continue a line of development.

As stated in the introduction, the advent of parallel computing and the challenge of developing scalable, high-performance codes have been the primary drivers mobilizing the community to seek a common solution. The achievement of the goals itemized above on scalable architectures currently available and with a view to future technological developments in high-performance computing is one of the key challenges we face.

A second key challenge is to meet these goals for a broad and very diverse community. To be a true community resource, the ESMF must be usable or adapatable to a wide variety of applications. It must satisfy the performance requirements of operational centers, as well as the flexibility and "usability" requirements of reserach organizations; it must be adequate for implementing large applications at govenment laboratories and for the smaller applications of individual university researchers; it must be usable with high-resolution, short-time (weather) and low-resolution, long-time (climate) modeling applications; and perhaps most challenging, it must serve both the modeling and data assimilation communities.

This diversity of applications of the ESMF is what is required to support the national earth science objectives and more specifically to support NASA's mission as outlined in the Earth Science Strategic Enterprise Plan 1998-2002 cited in the Announcement. NASA's mission in the earth sciences focuses on data: its acquisition, dissimination, analysis, interpretation, and use to improve our knowledge of the earth system and our ability to predict it behavior. An ESMF would make major contributions to NASA's activities in the last three of these.

Today, the analysis of geophysical data relies heavily on detailed and comprehensive numerical models of all components of the earth system and on sophisticated data assimilation algorithms capable of synthesizing myriad observations into a physically consistent picture. Further use of this information to predict the state of the system days or months in advance, or how it will respond to external perturbations years or decades from now also requires comprehensive models of the atmosphere, ocean and other components. NASA maintains efforts in all of these areas that would be served by the ESMF. In particular, work at Goddard's Data Assimilation Office and NASA's Seasonal to Interannual prediction project, both participants in this proposal, would benefit immediately from the development of the ESMF.

[Say something about each of the other institutions, emphasizing that this is not just a "critical mass" but includes all major modling and data assimilation efforts in the country]

[Plan for delivery. This seems like a strange place to put this, but that is what they ask for. I guess our position can be that adopation by our group itself consists of a major "delivery" to the bulk of the US community. We need to say something about bringing in smaller efforts as the prototype is tested. We can also put here something about documentation and maintainance of software repositories for both the framework and for applications developed with it.]

2.2 Applications

[Need a bit on the applications served by the ESMF, maybe we can steal some text from Parts II and III?]

3 Technical Plan

The community collaborating on this proposal has already made significant progress in defining the structure of the ESMF. In June 2000, each participating group agreed to prepare a strawman document describing their application requirements, ESMF scope and architecture, and implementation strategies (see http://www.scd.ucar.edu/css/NASACAN.htm). The initial degree of convergence was high, and a combined strawman was created. Collaborators met at NCAR in August 2000 to review and refine the combined strawman, and work on a more detailed design document for ESMF began. This work is ongoing.

This section describes some of the initial conclusions of our community effort to date.

3.1 ESMF Functionality

3.1.1 Scope

The scope of the ESMF must be extensive enough to offer significant advantages to the earth science community but modest enough to be completed with the resources offered through this Cooperative Agreement. Working from the design goals listed in the Scientific Rationale Section, the participants in this proposal collectively decided that the ESMF should focus on providing basic non-scientific infrastructure: general utilities, including optimized I/O, and coupling services. Scientific codes, GUIs, optimized math libraries for purposes other than regridding, and differential operators are not part of the proposed work. These functions could be introduced to the ESMF in the future.

Some of the specific functionality desired for the ESMF is described below.

The coupling mechanism should perform regridding, interpolation and communication of gridded, distributed data. The data may represent multiple fields or a single field, may be in the same or different executables, may be in code segments executing synchronously or asynchronously, and may be distributed among nodes and/or partitioned among multiple threads.

The software necessary to support the above capability includes tools for describing a wide variety of grids and decompositions, and for performing high-level, collective manipulations of fields defined on those grids. The software for specifying decompositions should interface to a mechanism for performing dynamic load balancing.

Both the coupling mechanism and application codes would use utility routines. The basics include communication libraries, sychronization, I/O, performance profiling, time management, and signal and error handling.

3.1.2 Requirements

In addition to achieving the ESMF overall goals (interoperable components, code reuse, simpler code maintenance, ...) the ESMF code must conform to a set of functional requirements. These include:

Performance portability Portability and computational efficiency over a wide range of platforms are essential. The framework should not significantly degrade performance of an existing code written without the framework. Optimized performance on scalar architectures for moderate numbers of processors (16-500) is the highest priority. The ESMF design should not result in codes that are too resource hungry to run effectively on large numbers of low-cost machines.

Flexible usage The application writer should be able to choose how much or how little of the ESMF to use. For example, some application developers may want to use ESMF for coupling complete, integrated component models while others may want to build component models using lower-level tools or data structures provided by ESMF.

Extensive grid support ESMF must be able to couple components that are discretized on: logically rectangular grids (rectangular, displaced pole, staggered, stretched); reduced (cut-out) and regional grids; unstructured grids (e.g., land grids, observations); phase space grids (e.g., spectral, Fourier); nested and adaptive grids; and cubed sphere and icosahedral grids. In addition we require support for describing masked regions and halo regions.

High performance, multi-format I/O The ESMF utilities must support a generic interface for I/O of self-describing data in netCDF, binary, GRIB and BUFR data formats. Other formats such as the EOS HDF and ODF data formats may be added later. There must be support for high-performance parallel I/O.

Multiple language bindings ESMF utility and coupling software must be usable by applications written in C/C++, F77 and F90.

Other requirements These include appropriate error and signal handling, runtime configurability, and an efficient, low maintenance implementation (e.g., auto-documentation from code).

3.2 Background

Several rudimentary systems with capabilities similar to those desired for ESMF have been developed with the help of participants in this proposal (FMS[reference?], GEMS[4], STAPL[2]). Each of these offers support for data parallelism within components plus a higher level of large-scale task parallelism for coupling components. Each of these also offers a set of data constructs and general utilities.

FMS, from NOAA/GFDL, and GEMS, from NASA/GSFC-NSIPP, are modular F90-based frameworks developed for use within GCMs and for coupled climate modeling. They have achieved institutional support and modest user bases but do not have the flexibility needed by the broader earth science community, as identified in the goals and requirements laid out for the ESMF. Specifically, these systems lack: integrated support for multithreading, which is likely to be a prerequisite for optimal performance; the ability to represent nested grids; highly optimized regridding, especially for unstructured grids; multi-format, high performance I/O; the ability to couple multiple executables.

STAPL, a object-based, C-based framework from MIT Lincoln Laboratory, offers a service for coupling SPMD components similar to FMS and GEMS as well as highly optimized communication for hybrid SMP-cluster type architectures. However, it was developed for the signal processing domain and its data structures and transforms are quite different than the gridded fields typical of earth system models.

Members of our collaboration have investigated popular frameworks, such as POOMA and Overture, and have found none that could be easily extended to support the high-performance, flexible coupling capabilities we require. Recently, a DOE/NCAR ACPI project involving several of our co-investigators undertook an investigation of current earth system couplers in the process of redesigning the NCAR CCSM coupler. Again, no high-level package was found that appeared to be an appropriate basis for the new coupler.

We have, however, identified a multitude of low-level utilities that could be extended or directly incorporated into the ESMF, such as documentation generators, timing libraries, and communication libraries. [mention WRAPPER, SCRIP] Specifically we plan to use many of the low-level utilities in PETSc, a widely used high-performance parallel library developed by one of our collaborators.

3.3 Architecture

3.3.1 Interoperability and reuse

Object-oriented design and design reuse - both standard framework techniques - are the methods we will use to achieve interoperability.

We will adopt an object-oriented, layered approach to the ESMF architecture. Object-oriented design enhances code reuse since a class structure encourages well-defined, general purpose, encapsulated code segments that can work in a variety of contexts. The ESMF will contain a robust toolkit of non-scientific software classes and functions that will be used to build more complex classes in the framework and can also be used to compose scientific applications.

A framework implies that overall design - how classes interoperate - is reused as well as code. In the ESMF design we propose, certain classes within an application will be provided by the framework and others will be supplied by the application developer. The classes provided by the developer must possess a core set of methods in order to interoperate within the framework. For example, an ocean model component of a climate model might need to contain a method that returns a description of its data grid and distribution. This information would be used by a generic coupling mechanism to understand how data should be routed to and from that ocean component.

Most components supplied by the developer will be scientific code segments that need to be coupled to other scientific components for an application. Another way of viewing the ocean model example is that the developer is specializing a framework component class by adding to it the additional data and methods associated with the ocean component.

The related features of object-orientation and design reuse are critical for increasing code interoperability. By establishing a standard set of methods and coupling interactions, researchers across the country will be able to to prepare codes that they can be confident will operate with a large set of others (of course, there is no guarantee of scientific compatability!)

3.3.2 Performance portability and ease of use

We wil achieve performance portability and ease of use by layering code.

Like object classes, layers help to organize code, though at a larger scale. The ESMF software will consists of five distinct software layers, each with well-defined functions and interfaces. The "lowest" layers are non-scientific and general in function. The "highest" layers describe functions that are more specific to the earth system domain. Methods in higher layers call methods from lower layers, and classes in higher layers are often aggregates of classes found in lower layers.

Machine-dependent code will be isolated to the lowest level in the framework by wrapping it in a generic interface. This improves performance portability by reducing the scope of changes that need to be made to run on or optimize for a new architecture, and allowing us to optimize primitive operations by using vendor-specific libraries.

Likewise, calls to higher-level communication functions such as transposes are isolated to a layer in the framework so that an application developer does not need to manage the details of distributed data transfers. This makes the application code easier to write and use.

3.3.3 Preliminary architecture

The five layers in the ESMF architecture are shown in Figure 1.

At the bottom are **low-level utilities**, such as communication primitives, error handling, timing, machine primitives (e.g., nodeId), and basic I/O operations. These are often machine-dependent and may be coded procedurally for efficiency.

The second layer is a set of **parallel utility classes**. These include a layout class that describes the portion of a machine over which a data object is distributed, and can associate a simple topology with it. The layout class includes a set of communication methods based on its indices.

Classes representing **distributed grids** are in the third layer. Distributed grids contain a layout and a grid specification that describes the grid coordinates and connectivity.

Fields are in the fourth layer. Fields contain a distributed grid, and a field specification that describes attributes related to the physical field (e.g., name, units). We will also support a class of field sets, for bundling fields that are discretized on the same distributed grid.

The **coupling and components** layer includes large scale components, such as atmosphere and land models, and the classes used to simplify the transfer of data between them.

[add Figure and more explanation. Input from Balaji?]

3.4 Implementation Notes

3.4.1 Language

The ESMF must support F77, F90, C and C++ user interfaces. Since most component models are written in F90, ensuring that interface is efficient is the highest priority.

We are likely to write the ESMF in more than one language. We anticipate coding the low-level utilities of the framework mainly in C/C++, leveraging existing code that is in PETSc and extending it to include support for multithreaded environments.

An implementation study at the beginning of the proposed work will help to assess the relative advantages of writing upper levels of the framework in F90 or C++. The study will include prototyping key functions in both languages to see if there are significant efficiency, ease of use, or interoperability issues, as well as addressing trends in areas like compiler quality and staff availability. The study will be acelerated by using existing codes such as FMS and GEMS. Findings will be presented in a brief Implementation Report (see Software Engineering Plan).

3.4.2 Hybrid programming strategy

Shared memory clusters are an increasingy prevalent architecture, and the current trend is towards more processors per node. This makes it essential to communicate information within a node as efficiently as possible, and this may mean multithreading rather than relying strictly on message passing. Including the capability to distribute data over nodes and computation over threads in ESMF will position the library to adapt to a range of future platforms.

3.5 Application code: The CCSM Flux Coupler

A natural testbed application to demonstrate the capabilities of the ESMF is the flux coupler for a coupled climate modeling system such as the Community Climate System Model (CCSM). The flux coupler supplies a number of crucial functions to the CCSM, including: coordination of time-stepping and synchronization of the component models; interpolation of data from one component model's grid to another; time-averaging of incoming data from one component model for later exchange with another component model; merging of flux data from multiple component models for delivery to another component model; exception and error handling; history output. Clearly the coupler is an excellent testbed for the following ESMF capabilities: support for multiple coordinate grids, and on-line generation of transformation matrices to support interpolation between these grids; exception and error handling utilities; performance monitoring and profiling utilities; [Ed. note—there's plenty more].

Having chosen the CCSM coupler as a testbed for the core of the ESMF, it is necessary to characterize the current state of the CCSM coupler in terms of the following criteria: parallelism and performance portablility; flexibility and extensibility.

In terms of parallelism, the CCSM currently is a multiple load image mixed-mode parallel application;that is, there are separate executables for each of the component models, and for the flux coupler. In the original version of the CCSM, each of the separate executables used shared memory parallelism, with message passing between the component models and the flux coupler. As the CCSM has evolved, the individual component models now employ message-passing and/or shared-memory parallelism, but the flux coupler has remained a purely shared-memory parallel application. This current (as of October, 2000) version of the flux coupler is a fair and sensible baseline against which the ESMF will be compared to assess completion of the milestones outlined in this proposal.

In terms of flexibility and extensibility, the CCSM coupler is quite rigid. Currently, the choice of component model grids is limited to the combinations of model grids for which interpolation weights have been generated off-line using the SCRIP package. This puts the onus on the model user wishing to introduce new components with different grids to generate the appropriate interpolation weights. It also places the opportunities offered by adaptive grids beyond the user's reach. If a user wishes to add new fields to the coupler, this requires modification of the coupler's source code. Taken together, these two previous points make the addition of a new component model to the CCSM a major undertaking. [Ed. Note–can't put my finger on it, but I think there's more worth adding here].

During the next year, the CCSM coupler will undergo considerable changes as the result of the DOE/NSF Accelerated Climate Prediction Initiative (ACPI) "Avant Garde" project. The goal of this project is to enhance the CCSM software in terms of performance portability, modularity, flexibility, and extensibility. The expected impact on the CCSM coupler is that it will be re-implemented in a more modular fashion using components from a "model coupling toolkit" that is currently under development. It is expected that this approach will make the coupler far more flexible and extensible. The toolkit under development will also make the resulting CCSM coupler a parallel application capable of exploiting the hybrid parallelism offered by the parallel platforms currently available to the U.S. scientific community. This project, however has a fairly limited timeframe of 18 months, and it is doubtful that it can realize the potential gains outlined in this proposal. [Ed. note–don't have quite the right words at hand to point out I doubt we'll be able to generate regriddings on-line at the end of ACPI -AG...just so much else to do first!].

We envision one or more of the following activities will demonstrate the efficacy of the ESMF:

- Re-implentation of the CCSM coupler using components supplied by the ESMF to create a performance-portable, mixed-mode parallel application capable of exploiting message-passing parallelism between shared-memory nodes;
- Demonstration of improvements to the ACPI "Next Generation Coupler" through substitution of ESMF components for those in the Model Coupling Toolkit.

3.6 Expertise in Scalable Grand Challenge Applications

[Note: I'm running an old version of LaTeX that doesn't seem to understand graphics packages like rotate. For now it was easiest to just rotate the plot from Max myself and jam it in, but the quality suffered – this will be fixed]

All participants in this proposal are heavily involved in the development of Grand Challenge earth science applications for massively parallel systems. Team members have been involved in this work for many years and have had experince with most MPP systems that have been produced over the last 15 years. Currently all participating institutions have highly optimized parallel implementations of at least some of their production codes. Figure **??** shows speed-up curves for a sampling of these applications. These were chosen to highlight the breadth of applications and platforms being used. Shown in the figure are results for both atmospheric (NCEP, NSIPP, CCM) and ocean (POP) models. Two of the models are spectral and two grid point; two are operational codes (NCEP and NSIPP) and two are research codes (CCM and POP). We have chosen to show results on platforms from four of the leading U.S. manufactures: SGI, Cray, IBM, and Compaq. Most of these models, however, run on multiple platforms. Additional results appear in Parts II and III.

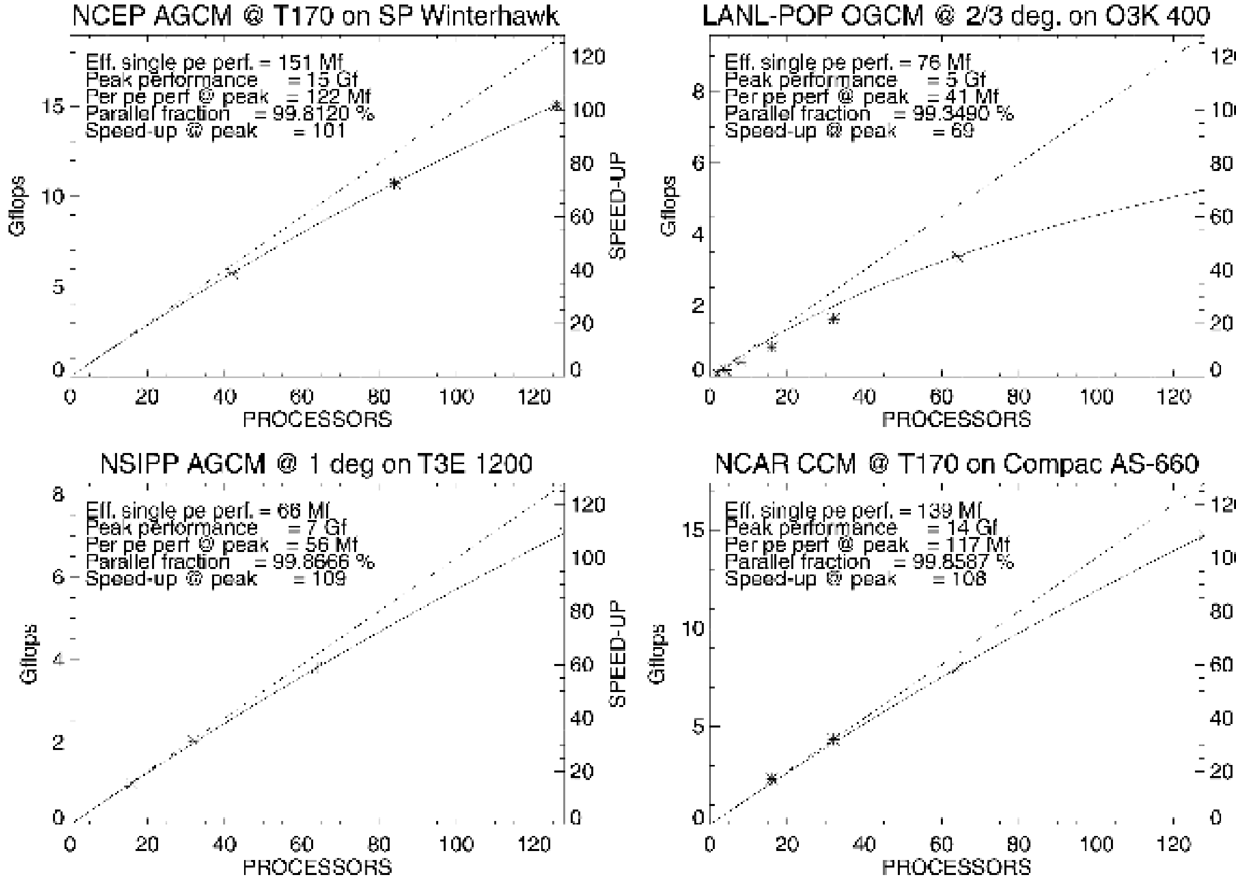

Figure 1: Speedup curves.

4 Management Plan

4.0.1 Management plan

The Principal Investigator of the proposal is Tim Killeen, the current Director of NCAR. He will serve as the primary contact and administrator of the proposed work. Dr. Killeen will negotiate agreements with NASA and among Investigator Team members, and will arrange for disbursement of funds after payment. He will supervise the overall activities of the Investigator Team and promote the ESMF project to the wider community.

Three oversight teams including the Co-Investigators of the project will closely track and guide the design and implementation of the ESMF software. The teams correspond to different aspects of the ESMF software (low-level utilities, fields and grids, coupling), and reflect different interests and expertise. All oversight teams consist of a mix of scientists, computer scientists and software engineers, with some individuals on multiple teams. Oversight teams will remain in close contact with an Implementation Team located at NCAR through regular teleconferences and meetings. The oversight teams will participate in requirements analysis, design reviews, prototyping and testing over the course of the project.

The Implementation Team will be managed day-to-day by a software engineering manager who will be responsible for establishing and maintaining appropriate software practices, drafting an Implementation Plan with the guidance of oversight teams, tracking the progress of the group against the Implementation Plan and revising the plan as needed, and verifying that unit and system tests are successfully passed. Additional responsibilities of the software manager are outlined in the Software Engineering Plan.

The key challenge in creating a management plan for the ESMF is to entrain broad expertise in the framework's development while insuring that work can proceed efficiently. We believe that the management structure we propose will do exactly this.

Our management plan will accomplish the following:

- engage a large community in the specification of requirements and the overall design of the framework to maximize expert input and user buy-in;
- utilize groups with more focused interests to oversee the design and implementation of specific framework components in order to achieve timely, informed decisions;
- delegate much of the work of design drafts, prototyping and production coding to a closely integrated, central team of software engineers;
- resolve inconsistencies and differences of opinion throughout the project by allowing final software engineering decisions to be made by the software manager and oversight team leaders.

4.0.2 Investigator team

[Still need to write this part.]

(II) Software Engineering Plan

In this section we present a software engineering plan for the ESMF, including software team structure and management, a software process that extends from system specification through distribution and maintenance, and tools and techniques to support development, collaboration, and distribution. Issues primarily relating to ESMF design, such as strategies for interoperability and reuse, are described in the Technical Section.

1 Software Teams and Management

An *implementation team* will be established at NCAR consisting of five software engineers supervised by a *software engineering manager.* One of the software engineers will be an *integrator* who is responsible for overall functions such as configuration management and defect tracking. This team will draft design specifications, prototype and implement ESMF components, test and validate the framework, and distribute releases. Some members of the implementation team will be retained at NCAR after the end of the project for ongoing development and support of the ESMF.

Software development will be guided by three partially overlapping *oversight teams* focused on different aspects of the framework: utilities, fields and grids, and coupling. The members of the oversight teams will include the co-investigators of this proposal, and will consist of a mix of software engineers, scientists and computer scientists. Each team will designate a lead. Responsibilities of the oversight teams will include reviewing design and tracking software implementation.

The software engineering manager and integrator will maintain a system view and ensure that development is coordinated. Oversight team leads and the software engineering manager will be charged with final decisions on split issues. [add dispersed application team, role of NASA integrator]

2 Software Process

The development team will follow a structured software process commensurate with CMM Level II [5, 6]. The process will include many of the procedures recommended by the Software Best Practices Initiative [1], such as software interface specification before implementation. Our documents and reviews will be simplified versions of those described in standard references [3, 7, 8]. We will aim for an effective process free of unnecessary overhead.

2.1 Staged Software Development

The major milestones described in Part V are the result of the coordinated completion of many smaller events, each of which has a "completion gate". These gates may be the approval of a document or the verification that a software test has been passed. Table 1 shows the progression of events in ESMF sofware development, and the product and gate associated with each event.

The initial set of events, labelled "ESMF Definition" is focused on specifying the ESMF system and procedures as a whole. Since we will be employing object-oriented design to enhance reuse and extensibility, during implementation our focus will shift to individual classes. The second group of events, "Class Implementation" describe the development steps applied to each class. As classes are completed they will be integrated into an evolving prototype of the ESMF. The final development stage, "Integration and Distribution", shows the integration of classes leading to a software release. The ESMF will have three major software releases corresponding to milestone I,J, and K; smaller releases and demonstrations are likely to be scheduled to insure that the project is on track.

Table 1 ESMF Software Event Progression

Event	*Product*	*Completion Gate*
	ESMF DEFINITION	
Requirements specification Brief document outlines ESMF functional scope and general requirements.	Requirements Document *Prepared by:* all collaborators	Document review *Reviewed by:* all collaborators
Architectural description Shows layering strategy and describes function and interaction of major components. Incorporates Requirements Document.	Architecture Document *Prepared by:* all collaborators	Document review *Reviewed by:* all collaborators
Software process definition Documentation of software procedures for the implementation team. This document will evolve as the project progresses.	Developer's Guide Document *Prepared by:* integrator	Document review *Reviewed by:* software manager
Implementation study Assesses existing software, optimal language, threading strategy and other issues as identified.	Implementation Report *Prepared by:* implementation team with collaborator input	Document review *Reviewed by:* all collaborators
Software implementation and test plan Plan for class implementation and testing based on milestones, class dependencies, and existing code.	Software Implementation and Test Plan *Prepared by:* software manager	Plan review *Reviewed by:* all collaborators
	CLASS IMPLEMENTATION	
Class design Includes requirements, function, and interface specification. Requirements may be reviewed first for complex classes. Groups of classes may be reviewed together.	Class Design Document *Prepared by:* class developer(s)	Design review *Reviewed by:* oversight team, software manager
Class implementation A class may be stubbed or only partially implemented for a given release.	Prototype code *Prepared by:* class developer(s)	Code review *Reviewed by:* oversight team, software manager
Class unit test A class is tested stand-alone with a variety of inputs.	Unit test code *Prepared and tested by:* class developer(s)	Unit test *Verified by:* software manager
	INTEGRATION AND DISTRIBUTION	
Class integration Unit tested class is integrated into an evolving prototype of the ESMF.	ESMF system prototype *Prepared by:* class developer(s), integrator	System test *Verified by:* software manager
User documentation updated Class design documentation is updated and converted to user documentation.	User's Guide *Prepared by:* class developer(s), integrator	Review before software release *Reviewed by:* software manager
System release Code and documentation is released. Requests for features and defects are tracked and incorporated into future releases.	System test *Prepared by:* integrator, software manager	ESMF system release *Reviewed by:* wide community

[Note chart needs some clarification wrt systm test and release. Also the format could use some improvement ...]

2.2 Documents, Reviews and Verification

Documents All documents will be prepared in a format that easily generates both hardcopy and web-friendly html. We intend to use a documentation generation tool such as doctext to automatically create and update portions of our documentation. Most documents will be placed under version control. We will structure design documents so that they can be easily converted to user documentation.

Reviews A variety of reviews will be held, including requirements, design and code reviews. The outcome of a review will either be a pass or, if significant changes are required, the scheduling of another review. The software engineering manager will attend all reviews for coordination.

Verification Class and system tests will be designed so that they verify that the code being tested fulfils its requirements.

3 Software Tools and Techniques

The following are some of the tools for development and collaboration that we plan to employ:

Configuration management We will likely use CVS for configuration management since it is mature, freely available, and the current community standard. We anticipate maintaining code at the following acceptance levels: *Active* (untested), *Unit tested*, and *Integrated* (code is part of a working ESMF prototype). These levels reflect the completion gates applied to code development shown in Table 1.

Software metrics The software engineering manager will track a simple set of software metrics throughout development in order to evaluate progress towards milestones and predict development schedules. These metrics will include LOSC for each code acceptance level and software engineer, and hours per software engineer spent on the project.

Defect tracking An open source tool such as Bugzilla will be used to maintain a database of defects and new feature requests. We will begin to use such a tool after a simple working prototype of the framework has been delivered.

Collaborative tools We plan to employ weekly teleconferences to keep oversight teams in closely in touch with the implementation team, and quarterly face-to-face meetings. We will continue to maintain a mailing list and may explore using forum software.

4 Source Availability and Distribution

We plan to develop our code in an open source development environment such as SourceForge. In order to meet milestones systematically, we will need to follow a well-defined and carefully managed software engineering plan. Until a complete prototype of the framework is delivered, we do not feel that an open participation model is appropriate. After it is released, we plan to engage the earth science community in contributing to the ESMF. Community improvements will be checked and formally added to next releases through a maintenance team at NCAR.

Source code and documentation will be distributed via an ESMF website. The website may be an extension of that currently maintained by the CMIWG. We plan to hold a series of workshops to introduce the broader community to the ESMF.

[To include: Developing and implementing an audit process, Collaboration plans with ESS Evaluation team]

(III) References

References

[1] Airlie Software Council. Principal best practices. Technical report, Software Program Managers Network.

[2] DeLuca, C., Heisey, C.W., Bond, R., and Daly, J.M. A portable object-based parallel library and framework for real-time radar signal processing. In *Scientific Computing in Object-Oriented Parallel Environments*, Berlin, Germany, 1997. Springer-Verlag.

[3] McConnell, S. *Code Complete*. Microsoft Press, 1993.

[4] Office Note Series on Global Modeling and Data Assimilation. Geos-3 data assimilation system. Technical Report DAO Office Note 97-06, NASA Goddard Space Flight Center, July 1997.

[5] Paulk, M.C., Curtis, W., Chrissis, M., and Weber, C.V. Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-24, DTIC Number ADA263403, Software Engineering Institute, February 1993.

[6] Paulk, M.C., Weber, C.V., Chrissis, M. Garcia, S.M., and Bush, M.W. Key practices of the capability maturity model, version 1.1. Technical Report CMU/SEI-93-TR-25, DTIC Number ADA263432, Software Engineering Institute, February 1993.

[7] Software Engineering Research Laboratory Series. Manager's handbook for software development. Technical Report SEL-84-01, NASA Goddard Space Flight Center, November 1990.

[8] Software Engineering Research Laboratory Series. Recommended approach to software development. Technical Report SEL-81-305, NASA Goddard Space Flight Center, June 1992.

(IV) Biographical Sketches

Curriculum Vitae of Barry F. Smith

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844

http://www.mcs.anl.gov/~bsmith   bsmith@mcs.anl.gov

Major appointments

Argonne National Laboratory, June 1994 – present

Computer Scientist, Mathematics and Computer Science Division, August 1995 – present

Assistant Computer Scientist, Mathematics and Computer Science Division, June 1994 - August 1995

Wilkinson Post-doctoral Fellow, Mathematics and Computer Science Division, August 1990 – August 1992

University of California at Los Angeles, August 1992 – June 1994

Computational and Applied Mathematics Assistant Professor (Visiting)

Education

New York University, September 1986 – August 1990

Ph. D. in Mathematics, September 1990

Yale University, September 1982 – May 1986

B. S. in Mathematics, May 1986

Honors and awards

Co-winner, Householder Prize for best dissertation in numerical linear algebra during the preceding three years, June 1993

Second Prize, Fifth Leslie Fox Prize Meeting, international prize in numerical analysis offered every two years, June 1991

First Prize, Student Paper Competition, Copper Mountain Conference on Iterative Methods, April 1990

Related publications

1. *On the Interaction of Architecture and Algorithm in the Domain-Based Parallelization of an Unstructured Grid Incompressible Flow Code*, D. K. Kaushik, D. E. Keyes and B. F. Smith, 1998, Proc. 10th Intl. Conf. on Domain Decomposition Methods, J. Mandel, et al, eds., Amer. Math. Soc., pp. 311–319.

2. *An Interface for Efficient Vector Scatters and Gathers on Parallel Machines*, Preprint ANL/MCS-P701-1197; submitted to International Journal of High Speed Computing.

3. *Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries.* S. Balay, W. Gropp, L. Curfman Mcinnes, and B. Smith, Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset and H. P. Langtangen, Ed. Birkhauser Press, 1997. pp 163–201.

4. *The Transition of Numerical Software: From Nuts-and-Bolts to Abstraction*, SIGNUM (ACM Special Interest Group on Numerical Computing) Newsletter, January 1998.

5. *Prospects for CFD on Petaflops Systems*, D. Kaushik, D. Keyes, and B. Smith, to appear in CFD Review, Wiley Press, 1997.

6. *Scalable, Extensible, and Portable Numerical Libraries*, W. Gropp and B. Smith, Proceedings of Scalable Parallel Libraries Conference, October 6-8, 1993, Mississippi State University, IEEE Press, pp. 87-93.

7. *PETSc 2.0 Users Manual*, S. Balay, L. Curfman McInnes, W. Gropp, and B. Smith, Argonne National Laboratory Report ANL-95/11.

(V) Milestones, Schedule and Costs

The dates on these milestones assume that the first payment is received June 2001.

Please note that the interoperability milestones H, I, J and K are identical to those in the other two proposals in a three-proposal set.

Milestone Number	**Milestone Label**	**Milestone Title**	**Expected Completion Date**	**Advance Payment Amount**
1	A	Software engineering plan completed *Deliver draft Developer's Guide specifying software procedures and conventions.*	Sept. 2001	$
2	E	Code baseline completed *Baseline CCSM Flux Coupler on ESS testbed in fully coupled CCSM configuration using stub component models. Metric will be time to solution, [others]. Documented source code delivered via the Web.*	Nov. 2001	$
3	H	Design policy for interoperability and community delivery agreed on *Deliver Architecture Document, Implemention Report, Draft Software Implementation and Test Plan. Prototype code used in the Implementation Study will be delivered via the Web.* *Identify a broader community interested in reviewing/testing the ESMF and mechanism for feedback.*	Dec. 2001	$
4	B	First Annual Report delivered *Submit FY01 Annual Report via the Web.*	June 2002	$
5	F	First code improvement completed *Demonstrate enhanced functionality: SPMD/MPMD component coupling* [Finish this specification.]	July 2002	$
6	I	Interoperability prototype from Milestone "H" tested with improved codes *Prototype implementation of 1 of 5 modeling applications using the framework: CCSM, WRF, GFDL, NSIPP, mitGCM. May be partial coupling/stubs.* *Prototype implementation of 1 of 3 data assimilation schemes using the framework with model stub. Source code delivered via the Web.*	Nov. 2002	$
7	C	Second Annual report delivered *Submit FY02 Annual Report via the Web.*	June 2003	$

(continued)

Milestone Number	**Milestone Label**	**Milestone Title**	**Expected Completion Date**	**Advance Payment Amount**
8	G	Second code improvement completed [Finish this specification]	July 2003	$
9	J	Full interoperability demonstrated using improved codes *Full implementation of 2 of 5 modeling applications using the framework: CCSM, WRF, GFDL, NSIPP, mitGCM.* *Full implementation of 1 of 3 optimized asimilation schemes and models.* *Using the framework internally within models/assimilation schemes as well as for coupling components.*	Nov. 2003	$
10	K	Customer delivery accomplished *Port the framework to a new platform within 1 month and demonstrate its operation. Deliver a user-friendly website where the ESMF source code and User's Guide can be obtained, and hold a workshop on framework usage.*	May 2004	$
11	D	Final Report delivered *Submit Final Report to ESS via the Web.*	June 2004	$

(VI) Endorsement Letters

Stub

(VII) Education and Public Outreach

Stub