AIAA-98-4740

FRAMEWORK REQUIREMENTS FOR MDO APPLICATION DEVELOPMENT

A. O. Salas* and J. C. Townsend†
NASA Langley Research Center, Hampton, VA 23681-2199

Abstract

Frameworks or problem solving environments that support application development form an active area of research. The Multidisciplinary Optimization Branch at NASA Langley Research Center is investigating frameworks for supporting multidisciplinary analysis and optimization research. The Branch has generated a list of framework requirements, based on the experience gained from the Framework for Interdisciplinary Design Optimization project and the information acquired during a framework evaluation process. In this study, four existing frameworks are examined against these requirements. The results of this examination suggest several topics for further framework research.

Introduction

Multidisciplinary design of aerospace systems is a complex, computationally intensive process that combines discipline analyses with design-space search and decision making. The decision making is based on engineering judgement but is greatly assisted by computer automation. Because the point of view, design emphasis, and design approach of discipline specialists can be quite different, the practice has often been for each discipline to be optimized independently, having limited direct interaction or communication with other disciplines. The present aim of Multidisciplinary Design Optimization (MDO) is to meet the needs for increased interdisciplinary interaction and communication and for reduced design cycle time.[1]

The development of computational frameworks or problem solving environments offers the capability to meet these needs via the use of sophisticated computational procedures combined with state-of-the-art optimization or design improvement techniques. Specifically, the development of computational frameworks to assist in rapid generation of "what-if" scenarios with minimal programming effort would be a powerful aid to the designer in improving the results of the design process and in reducing the time and thus the costs.

The Multidisciplinary Optimization Branch (MDOB) at NASA Langley Research Center (LaRC) recognizes the need for a framework that supports MDO applications—in particular, a framework that can support the implementation and execution of MDO applications and can provide a set of support services commonly needed in applications of this type. By having the framework support the integration of various processes of the MDO application, the designer would be able to concentrate more on the application and less on the programming details. The framework should automate the integration activities, thereby eliminating the hurdles otherwise present when transferring data among processes. In addition to development and execution support, a framework could provide a common working environment, which would increase the productivity of multidisciplinary projects.

The MDOB has participated in the development of the Framework for Interdisciplinary Design Optimization (FIDO),[2] sponsored by the High Performance Computing and Communication Program (HPCCP). The purpose of the FIDO project is to investigate the use of a distributed, heterogeneous computing system to facilitate communications, apply computer automation, and introduce parallel computing to produce a truly multidisciplinary process. This framework is intended to demonstrate technical feasibility and usefulness for selected applications and to provide a working environment for use by LaRC researchers testing various optimization schemes. It automates the coordination of analyses by the various disciplines (each on its assigned computer) into an integrated optimization scheme, while allowing for visualization and steering by the designer.

* Research Scientist, Fluid Mechanics and Acoustics Division (FMAD), MDOB
† Senior Research Scientist, FMAD, MDOB, Associate Fellow AIAA

Copyright @ 1998 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the Copyright Owner.

Increasingly complex multidisciplinary models of the High Speed Civil Transport (HSCT) have been implemented in FIDO. The framework was first demonstrated for a version of this design problem with fast, limited-fidelity discipline codes (equivalent plate structural analysis, linearized aerodynamic analysis, propulsion table lookup, and a simple range equation for performance fuel weight estimation), a geometry given by a set of points, a small number of design variables (on the order of ten), and a simple objective function. Recently the HSCT application has been demonstrated with medium-fidelity structural (coarse-grain, finite-element analysis) and aerodynamic (marching supersonic Euler) codes coupled in a static aeroelastic loop. A redesigned HSCT application, now in progress, will provide the additional realism afforded by full nonlinear aerodynamic corrections, realistic finite-element analysis and weights estimation, full mission-cycle performance evaluation, and an actual, proposed HSCT geometry with realistic constraints, such as ground scrape.

The major limitation with FIDO is that the system was implemented with the purpose of demonstrating a specific application—the HSCT design. As a consequence, the sequence of processes is hard coded, making it difficult to modify. The intertwining of the framework tools and the application formulation, coupled with a lack of documentation, has made FIDO inaccessible for use by researchers who did not participate closely in its development.

Although FIDO was not implemented as a generic framework for MDO applications, its development has provided much experience with the issues of framework architecture and problem formulation. Because of FIDO's limitations described above and the amount of resources required to continue development and maintenance of a computational environment for MDO research, MDOB and HPCCP have been exploring alternative frameworks.

Within the past several years, MDOB and HPCCP conducted an evaluation process to investigate currently available frameworks. With the goal of developing a set of requirements for MDO frameworks, MDOB and HPCCP interacted with LaRC organizations and other government agencies involved in MDO research. Several candidate frameworks with the potential for supporting MDO research activities were identified.[3] The evaluation process relied on written information and personal contact with the framework developers; the evaluation did not include hands-on testing of these frameworks. Framework evaluation efforts continue in MDOB.

This paper provides conclusions made to date about the framework characteristics necessary for supporting MDO applications. First, the necessary framework requirements identified during the evaluation process are discussed. Next, a number of existing frameworks that appear to be relevant to MDO work are briefly described, and then several of these frameworks are described in more detail.‡ Because some time has passed since the evaluation process previously described, the set of frameworks mentioned in this paper does not coincide exactly with the frameworks originally investigated. The final section identifies some weaknesses found in current frameworks and suggests several topics for further research.

MDO Framework Requirements

The purpose of a framework is to provide support for multidisciplinary design optimization application development and execution. This section lists a set of requirements for an ideal framework to be employed in LaRC's MDO research. The requirements are presented from the following points of view: architectural design, problem formulation construction, problem execution, and information access.

Architectural Design

A framework should provide a Graphical User Interface (GUI) that is intuitive. The GUI should be designed such that the user can quickly learn to use the features of the framework effectively. Such a GUI would encourage the user to take advantage of the benefits offered by the framework.

A framework should be designed using object-oriented principles. Object-oriented design[4] has several advantages in MDO applications. For example, object-oriented principles allow switching of analysis or optimization methods at run time. In addition, object-oriented concepts extend naturally into distributed computing, which is moving in the direction of distributed object technology.

A framework should be extensible and should provide support for developing the interfaces required to integrate new processes into the system. The user should be able to integrate new discipline codes, optimization methods, and other tools of interest into the framework.

‡ The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

As a result, the user would avoid having to wait for the needed features to appear in new releases.

A framework should not impose an unreasonable amount of overhead on the optimization process. Naturally, there will be some reduction in speed when the user is not fine-tuning the code. However, the framework should provide some performance measurements so that the user can identify time-consuming activities.

A framework should be able to handle large problem sizes. Currently, a framework should be able to support problems with at least several hundred design variables. In the future, a framework should support problems with thousands of design variables.

A framework should support collaborative design. MDO involves the expertise of multiple discipline designers. The designers need to be able to conveniently work together on the problem. A framework architecture that allows simultaneous access to the problem data by multiple users is desirable.

A framework design should be based on standards. Examples of standards include message passing, database access, and languages. Use of standards preserves investment and results in lower maintenance costs.

Problem Formulation Construction

A framework should allow the user to configure complex branching and iterative MDO problem formulations easily without low-level programming. By raising the level of abstraction at which the user programs the MDO problem, problems could be constructed faster and be less prone to error. Ideally, a framework would provide a visual programming interface for connecting processes.

A framework should allow the user to easily reconfigure existing MDO problem formulations. Problem formulation reconfiguration examples include replacing existing processes with new ones, deleting processes, or adding new processes to the application. Replacing processes with other processes may be desirable when experimenting with different levels of discipline fidelity. Adding/deleting processes may be desirable when adding or removing disciplines to the MDO problem. For example, the user may want to delete a process if it is no longer generating useful data. Fast reconfiguration supports the user in exploring alternative views of the problem without having to build the new problem from scratch.

A framework should support the user in incorporating legacy codes (written in a variety of languages) and proprietary codes (where the source is not available) into the MDO problem formulation. A major purpose of a framework is to support code reuse. In order for the framework to enhance productivity, users must be able to continue the use of familiar codes with no code changes required. The framework should provide tools for creating wrappers that would generate the appropriate input files, invoke the discipline programs, and automatically extract the output of interest.

A framework should allow the user to integrate discipline analyses with several optimization methods, including multilevel schemes involving suboptimizations. Since no one optimization method is best for all problems, it is important that a framework support experimentation with different methods.[5,6] The user should be able to select a combination of optimization methods when defining the optimization problem.

A framework should provide facilities for debugging of multiple processes executing on computers across a network. The framework should provide feedback to the user when problems are not constructed properly. The user should be able to "step through" the application during execution, monitoring progress of several remote computations.

Problem Execution

A framework should automate the execution of processes and the movement of data. In the traditional mode of multidisciplinary design, engineers wait to receive data from another discipline and then reformat it for input to their discipline. A framework should eliminate this delay by automating the preparation of input files, the execution of disciplines and optimization methods, the extraction of data from output files, and the transfer of data between processes.

A framework should be able to execute multiple processes in parallel. For computationally intensive MDO problems, the user should be able to identify and take advantage of coarse-grain parallelism within the problem. For example, several discipline codes may be able to execute in parallel without affecting correctness of results. Furthermore, in the case of some multilevel optimization formulations, subsystem optimizations are able to proceed in parallel.

A framework should support execution distributed across a network of heterogeneous computers. The user should be able to take advantage of resources on the network and of codes that have been optimized for certain hardware.

A framework should support user interaction (steering) during the design cycle. Realistically, the user

needs to evaluate results as the execution progresses and adjust the problem as needed. For example, the user may want to substitute processes, disable execution of a process, or switch optimization methods. Also, the user may want to adjust the design variable and/or constraint sets.

A framework should allow the user to operate in a batch mode. For productivity, the user needs to be able to define a problem or set of problems which can be executed one after the other without any manual intervention. For example, the user could take advantage of this capability to experiment with multiple starting points for an optimization problem.

Information Access

A framework should provide database management features. For larger problems, it is convenient to have a central database for maintaining data used by multiple disciplines. The user should have the option of defining which data are written to and read from the central database.

A framework should provide the capability to visualize intermediate and final optimization and analysis results. Also, the user should be able to easily track histories of chosen design variables, behavior variables, constraints, and objective function values. These results should become available as soon as they are stored in the database and remain easily accessible after computations are complete.

A framework should provide a monitoring capability for viewing the status of an execution, including the system status. For example, a framework should provide visual feedback on which processes are currently executing. This feature would alert the user to potential problems in the system, such as failure of a computation to complete within a reasonable time.

A framework should provide some mechanism for fault tolerance. For example, if the computer on which a process is executing fails, the user should be able to recover from an earlier automatic checkpoint with little or no loss of data. Even if no problems occur, the framework should provide a restart capability so that the user can begin a problem from a previous state.

Research and Development in MDO Frameworks

There is much activity in the area of frameworks and/or problem solving environments in government labs, industry, and universities. This section provides brief descriptions of those systems which seem most relevant to the authors.

The following three existing commercial products provide optimization toolkit environments allowing the user to integrate analyses with optimization methods in a flexible manner. Each product provides GUI services for reviewing results of an optimization process.

- iSIGHT (Engineous Software, Inc.) - The iSIGHT[7] product provides an optimization toolkit that allows a combination of optimization methods [numerical, heuristic, exploratory, design of experiments (DOE), and response-surface modeling (RSM)] to be applied to the MDO application.
- LMS Optimus (LMS Numerical Technologies) - LMS Optimus[8] provides nonlinear programming optimization techniques as well as DOE and RSM methods.
- Pointer (Synaps, Inc.) - Pointer provides genetic, downhill simplex, and gradient optimization techniques.

A noncommercial framework that provides an optimization toolkit capability is DAKOTA (Design Analysis Kit for OpTimizAtion), developed by Sandia National Laboratories. Sandia has used DAKOTA to implement applications on massively parallel machines,[9,10] as well as on workstation clusters.

Several multidisciplinary environments that focus less on providing an optimization toolkit and more on exploiting a distributed, heterogeneous computing environment are listed below.

- FIDO (NASA LaRC) - FIDO[2] was developed to demonstrate distributed and parallel execution of a multidisciplinary analysis and optimization application using the HSCT as its example. The project is supported by HPCCP.
- NPSS [NASA Lewis Research Center (LeRC)] - NPSS[11] (Numerical Propulsion System Simulation), supported by HPCCP, enables multidisciplinary design and analysis of engines.
- Access Manager (Boeing) - Access Manager[12] supports multidisciplinary analysis and design in a distributed, heterogeneous computing environment. A key feature of this system allows the user to control the processes via a GUI.
- MIDAS (Jet Propulsion Laboratory) - MIDAS[13] (Multidisciplinary Integrated Design Assistant for Spacecraft) supports integration for multidisciplinary analysis in a distributed, heterogeneous environment.
- MDICE-AE (CFD Research Corporation) - MDICE-AE (Multi-Disciplinary Computing Envi-

ronment for Aeroelasticity) supports aeroelastic analysis calculations. This framework is based on a distributed object model.

- Phoenix Integration, Inc. - For integrating multidisciplinary problems, this company is applying the concept of analysis servers that remotely host analysis codes and client design tools that connect to the analysis servers. The servers provide tools for wrapping and error recovery.
- LAWE[14] [High Technology Corporation (HTC)] - HTC promises a programming environment that will support development of large, distributed applications using high-level communication objects. LAWE (Large Application Working Environment) will be composed of subsystems for communications, visual programming, input and output display, and system monitoring. It is being developed under a NASA Small Business Innovative Research contract.

Some design tool products have a stronger focus on the data involved in the design. Below are a few examples of these tools.

- AML (TechnoSoft, Inc.) - AML[15] (Adaptive Modeling Language) employs a unified part model paradigm and a demand-driven calculation feature.
- IMAGE (Georgia Institute of Technology) - IMAGE[16] (Intelligent Multidisciplinary Aircraft Generation Environment) is a research project of the Aerospace Engineering Department. A feature of this framework is the provision of object-oriented data management utilities for use during design processes. IMAGE also provides a distributed computing capability.
- DARWIN (NASA Ames Research Center) - The Analytical Tools and Environments for Design program[17] is developing information technologies for use in the design of aeronautical systems. As part of this effort, DARWIN[18] (Developmental Aeronautics Revolutionizing Wind-tunnels with Intelligent systems for NASA) aims to reduce design cycle time by improving access to experimental data.

Web technology appears to be ideal for achieving several of the framework requirements, such as facilitating collaboration among researchers and access to information. The DARWIN framework is an example of a system using Web technology. Users access data by means of Web browsers; the data returned is generated by CGI (Common Gateway Interface) scripts and visualized via graphing Java[§] applets. Another project exploring the use of Web technology was conducted by MDOB at NASA LaRC.[19,20] This framework combines the use of a knowledge-based system for determining the processes ready for execution and Web technology for controlling processes, monitoring execution status, and visualizing problem data.

Several research projects (e.g., Legion,[21] Globus[22]) are focusing heavily on distributed computing technology and are tackling complex issues such as security, fault tolerance, and resource management. MDO is one of the application areas that will benefit from this research.

MDO Frameworks

This section describes in more detail four of the frameworks mentioned above that support MDO: FIDO, iSIGHT, LMS Optimus, and DAKOTA. The amount of in-house experience obtained with FIDO warrants its description here. The evaluation process described in the "Introduction" section revealed the iSIGHT and DAKOTA frameworks as two of the most relevant MDO frameworks available. Information acquired since the evaluation study reveals LMS Optimus to be another promising framework.

The descriptions that follow identify some of the major features supported by each framework and some of the requirements, as discussed in a previous section, that the frameworks are known to support. Due to lack of information or lack of experience with a particular framework, not all requirements can be addressed for each one.

FIDO

The FIDO project was briefly discussed in the "Introduction" section. This section describes the FIDO features in more detail.

Architectural Design. The FIDO architecture is modular. The framework is organized into distributed computational and service modules, which communicate through a communications library.[23] There is a computational module for each discipline contributing to the application. The service modules, such as the GUI, Executive (control), Data Manager, Setup, and Spy, are intended to be application independent.

The communications library contains functions designed to facilitate communications among a general system of computer codes executed in a heterogeneous, distributed network of computers. This library allows FIDO to be programmed without directly accessing the

§ Java is a trademark of Sun Microsystems, Inc.

underlying, message-passing primitives and minimizes the impact on FIDO due to any changes in them. Currently, the PVM (Parallel Virtual Machine) primitives[24] from the Oak Ridge National Laboratory are used.

The GUI is limited to displaying the status of the execution. The Spy tool promotes collaboration among researchers by allowing access to Spy from multiple remote computers.

Although object-oriented principles were not applied to FIDO, there was a strong emphasis on producing a modular system. As the complexity of the HSCT demonstration problem increased, more attention was given to defining discipline interfaces so that disciplines with differing fidelity levels could be interchanged.

The development of persistent discipline drivers has increased execution efficiency. Persistent drivers allow data that are to be shared among related codes within a discipline to be conveniently held in memory. Timing routines inserted into the application allow the user to determine the amount of execution time for various parts of the computation.

Problem Formulation Construction. A major limitation of FIDO is that it lacks support for building and reconfiguring MDO problem formulations at a higher level of abstraction than coding in the currently available programming languages, such as FORTRAN and C. Some discipline codes used in FIDO are decades old and originally contained deeply embedded *print* and *stop* statements. These codes were modified to behave as library subroutines and are invoked from the appropriate discipline driver. As a result, the discipline driver and the associated discipline codes are linked into one executable program. Overall, this is not a desirable approach because it involves extra work, duplicates maintenance tasks, and does not promote code reuse.

Coordination of discipline analyses is provided by a problem-dependent Master module. Currently, the code for this module must be rewritten for each specific MDO application. For example, the initial focus application of the FIDO project has a 1000-line C-code Master module to perform the complex iterative looping behavior required for even the simplified preliminary design of an HSCT.

Only gradient-based optimization methods have been used within FIDO. In the early versions of FIDO, CONMIN[25] was incorporated into the HSCT application. Later, the optimization module was modified to include KSOPT[26] as an alternative to CONMIN. The user identifies the choice of optimizer by a data file parameter.

The various modules were compiled with debugging options so that the code could be stepped through during execution.

Problem Execution. The user designs the FIDO Master module so that the optimization and analysis processes are invoked and synchronized appropriately. The user provides the synchronization logic within the discipline drivers in the form of calls to the communications library's send and receive routines.

The FIDO Setup module allows the user to choose the system configuration and the initial conditions and constraints of the optimization process from a range of previously defined possibilities. These are contained in four configuration files that define the data in standardized formats.

All major data elements (individual items or file pointers) that are shared between modules are passed to, stored in, and retrieved from the central Data Manager. Using file pointers, data files are passed directly on request from the generating computer to the requesting computer. Because the communications library allows direct passing of data messages between the discipline computers, direct communication of messages can be implemented if the increased efficiency warrants it.

The discipline drivers and their corresponding analyses are assigned to execute in parallel on different computers defined to be part of the PVM network. Because the discipline codes have short execution times, the parallelism exploited thus far in the FIDO HSCT applications is mainly in the calculation of derivatives using a finite-difference technique.

From the beginning of its development, FIDO was designed to allow some interactivity during the design cycle. This feature is accomplished using the Spy tool, which allows the user to steer the process while the application is executing. By means of the Spy tool, the user may change current values of design variables, constraints, and parameters. On the other hand, FIDO lacks a convenient way of setting up multiple problems that can execute one after the other. In particular, changing the initial conditions of a problem requires manually editing the input and configuration files.

Information Access. The FIDO Data Manager allows storage and retrieval of data during problem execution and is designed so that no additional coding is required for new problems. The user must define the data to be handled prior to execution.

The FIDO Spy module allows the user to access and plot data from previous design cycles. The accessible data includes information on the cycle status and

selected scalar and array data from each cycle. The data can be displayed as text or graphics. However, the database is not persistent, so FIDO must be running for data to be accessed.

The FIDO GUI displays the state of the problem execution at all times. The GUI displays the problem formulation and uses color to indicate those processes that are starting up, executing, inactive, or shutting down.

Although FIDO allows restart from a completed optimization cycle, it provides no other fault-tolerance capability. However, a restart requires some data file preparation.

iSIGHT

The iSIGHT framework is a generic shell environment for supporting multidisciplinary optimization. A key feature of iSIGHT is the ability to combine numeric, exploratory, and heuristic methods during an optimization.

Architectural Design. The iSIGHT environment consists of several modules including an interpreter, toolkits, and GUIs. The Tcl[27] language is the interpreter that provides the "glue" for integrating various processes. GUI services are provided for connecting processes, defining the optimization plan, and monitoring results.

GUI services are provided for wrapping discipline codes. In addition, the user may integrate additional optimization techniques into iSIGHT. However, the iSIGHT Application Programming Interface (API) must be used to create the appropriate interface between the optimizer and the framework. In addition, a Tcl command must be created for the optimization technique.

Problem Formulation Construction. The iSIGHT framework provides the user with both a GUI, in which modules are represented by icons, and the Multidisciplinary Optimization Language (MDOL) for constructing MDO problems.[28] Use of the GUI to define the problem generates the appropriate MDOL file, referred to as a description file. MDOL has a block structure style and English-like language constructs.

The GUI provides the user with building blocks representing discipline codes and calculation blocks that may be needed in addition to the discipline codes. The user may define the input, output, and execution invocation of the discipline code blocks, as well as the arithmetic expressions for the calculations blocks. However, within the GUI, the user is limited to defining a sequential order for the disciplines and calculations. If the problem logic requires branching or iteration, the user must express this logic through a combination of Tcl and either MDOL or iSIGHT APIs in the description file. Modification to the problem formulation requires modification of the description file.

The iSIGHT framework allows users to construct MDO applications using existing discipline codes without modifications by interactively generating a code wrapper. Parsing utilities create the appropriate input files and extract the appropriate data from discipline output files. Using the GUI and the input and output file templates for a discipline code, the user can generate the appropriate file parsing commands. As a result, the user is able to integrate legacy and proprietary codes into the MDO problem.

Optimization techniques in iSIGHT include numerical, exploratory, expert system, and response surface methodologies.[29] The techniques or combination of techniques, as well as the design variables, constraints, and objective function, can be defined via the GUI or the MDOL description file.

Problem Execution. The iSIGHT framework automates the execution of the various discipline codes and calculation blocks, the handling of the data, and the adjustment of design variables during optimization. The computational processes defined in the problem formulation are executed sequentially, because iSIGHT provides no support for parallelism. There is very limited support for distributed computation. For example, a discipline code may initiate a remote process; however, all description codes for a problem must reside in the same directory on a single computer.

Interactive features[30] in iSIGHT provide the capability to pause the execution and continue it later. The user can stop the execution to modify the optimization methods, design variables, constraints, and objective function. Upon restart, the execution resumes from the best design point of the previous optimization. Discipline codes can be switched only if the appropriate logic is present in the description file.

Information Access. The iSIGHT framework lacks a database capability other than the data management toolkit that keeps a history of the design states. Therefore, data sharing among several discipline codes has to be accomplished by writing and parsing files.

A monitoring capability is provided that can be applied at any time during execution.[30] Input and output values can be monitored in tabular or graphical form. In addition, the user can review the data from a completed optimization and can restart the optimization process from a design point previously computed. The GUI also

indicates which process in a task is running by changing the appearance of the module icon.

LMS Optimus

Another framework for multidisciplinary optimization, LMS Optimus, allows a user to set up a problem, select a method to be used with the problem, and analyze the results.[31] Features of LMS Optimus provide nonlinear programming (NLP), DOE, and RSM techniques for optimization.

Architectural Design. The Optimus Kernel module contains the GUI, which provides features for constructing the analysis sequence and design problem and for analyzing the results. The GUI is written using the C++ language and Motif¶. The user can include any analysis code as part of an MDO application as long as the design input and output can be identified in the input/output files.

Two optimization modules are provided in LMS Optimus: the DOE/RSM module and the NLP module. A recently available feature allows the integration of an external optimizer. All that is required for integration of an optimizer is that it writes the adjusted design variables to a file and reads the analysis results from a file.

Due to internal array sizes, the maximum number of design variables allowed is 50; the maximum number of design outputs allowed is 200.

Problem Formulation Construction. The user employs the LMS Optimus GUI to define the analysis sequence. Through the GUI, the user identifies the analyses and their corresponding input and output files. In addition, the files associated with the design data input and output are identified. Actions taken through the GUI result in the creation of a command file. The command file contains sections for defining design inputs, design outputs, discipline input and output file parsing commands, analysis sequencing, and optimization method selection. The sequencing commands include *if/then/else* and *for* control statements. The GUI generates only a subset of commands that can be included in the command file; the user can edit the command file to include additional commands.

The LMS Optimus user can include legacy and proprietary codes in the analysis sequence without making any modifications. The GUI can be used to identify the design data in the input files; before each analysis is executed, the input files are automatically constructed by the framework to include appropriate input. Similarly, the user identifies, via the GUI, the output to be extracted from the output file; after the analysis completes execution, the data is automatically extracted.

Once the analysis is defined, the user can select either a user-defined table of experiments, an NLP method, or a DOE/RSM method to be integrated with the analysis. The NLP methods available include sequential quadratic programming and generalized reduced gradients. The results from a DOE method can be used to form an RSM. The RSM may then be used in place of the full analysis during an optimization.

Problem Execution. The Optimus Kernel automates the execution of the various discipline codes included in the analysis, manages the input and output data, and adjusts the design variables. The processes defined in the analysis are executed sequentially. For distributed computing support, the command language includes a command for executing a remote process.

Information Access. The results from a completed NLP or DOE method can be loaded by the GUI and postprocessed. The results of an optimization can be displayed in a tabular format. Several options exist for visually analyzing an RSM.

DAKOTA

The DAKOTA design provides a flexible and extensible interface between analysis codes and iteration methods. Methods are included for optimization, uncertainty quantification, parameter estimation, and sensitivity analysis.

Architectural Design. The DAKOTA design is based on object-oriented principles and is implemented with the C++ language. The definition of generic interfaces between optimization methods and analysis codes hides the specifics of each. Use of these interfaces and object-oriented language features promotes the “plug and play” capability.

Problem Formulation Construction. To define the MDO application, the user must create a file that specifies information about interfaces, variables, responses, strategies, and methods.[32] In DAKOTA, “strategies” manage methods and “interfaces” provide access to the discipline codes, which map the variables to the responses.

Several types of interfaces are defined in DAKOTA, the primary being the application interface. The application interface allows discipline codes to be accessed through either system calls or direct function calls. The direct function call interface requires converting main programs to function calls and linking the

¶ Motif is a registered trademark of Open Software Foundation, Ltd.

functions into the DAKOTA executable. The system call interface allows access to external programs; communication between the external program and DAKOTA is accomplish via files.

The interface section of the specification file must include the name of the analysis (or analysis driver), and if required, the names for the input and output filters (i.e. pre- and post-processors). Note that only one analysis driver may be specified; however, an entire MDO application, developed outside of DAKOTA, along with an input and an output filter, can be accessed through these three names. The input filter must use the design parameter list provided by DAKOTA to prepare the input for the analysis driver. Also, the output filter must retrieve data from the analysis driver and prepare the response and sensitivity data in the format required for use by DAKOTA.

A variety of optimization methods are provided, including NLP and genetic algorithms. The DAKOTA strategies manage multiple methods, disciplines, and approximations. The strategies include *single*, *multilevel hybrid*, and *sequential approximate optimization*. The *single* strategy allows a single method to be used with a single discipline. The *multilevel hybrid* strategy allows multiple methods to be used in succession with a discipline. This strategy uses the best solution from one method as the starting point for the next method. The switching criteria used can either be based on an individual method's convergence criteria or an adaptive technique that employs method performance metrics. The *sequential approximate optimization* strategy uses both a discipline and an approximation of the discipline. The approximation model is optimized, and the discipline model is evaluated at the approximate optimal solution. These results are used to update the approximation.

Problem Execution. Both the execution of the analysis driver and input/output filters and the transfer of data between these and the optimization methods are automated by DAKOTA. Distributed computing is supported using MPI message passing on workstation clusters and on massively parallel supercomputers. The asynchronous function evaluation command option allows concurrent analysis calculations and is available with both system call and direct function interfaces. This feature can be used when calculating derivatives using finite differences or when using the parallel algorithms provided in DAKOTA.

Information Access. There is an option for the user to specify creation of a restart log. Also, several options are available for handling application failure recovery.

Concluding Remarks

At LaRC, MDOB has gained experience in the development and use of frameworks that support MDO research. Framework evaluation and FIDO research activities in MDOB have generated a set of framework requirements. The FIDO, iSIGHT, LMS Optimus, and DAKOTA frameworks have been examined against these requirements.

None of the frameworks address all of the requirements; each has its strengths and weaknesses. Several major areas for future framework research include support for problem construction, distributed and parallel computing, database management, debugging, and designer interactivity. The least support in problem formulation construction is provided by FIDO. In the other three frameworks, support that allows existing codes to be integrated without modification typically is available via parsing tools and system calls. The LMS Optimus and iSIGHT frameworks provide visual programming support for simple formulations.

Although more database management, distributed and parallel computing capabilities are available in FIDO than in the others, the user must program at a low level to exploit these features. DAKOTA has more parallel and distributed capability than iSIGHT and LMS Optimus. Central database capabilities are not available in iSIGHT, in LMS Optimus, nor in DAKOTA. Visualization of optimization results is available during execution within FIDO and iSIGHT but only after execution within LMS Optimus.

Although FIDO does not meet all of the requirements discussed, the general architecture of the framework has proven its worth. The complex design processes that have been implemented in FIDO reveal the benefits of FIDO features, such as the Spy tool for promoting collaboration and design steering, the persistent discipline drivers for promoting efficiency, and the Database Manager for promoting data sharing.

Except for FIDO, LaRC's MDOB has more experience using iSIGHT than the other frameworks, having used it to implement an early version of the HSCT application. Based on this experience, it was decided that iSIGHT was not yet ready for use on the very complex, distributed, high-fidelity HSCT application currently being designed. A new version of iSIGHT, which is scheduled for release in the Fall of 1998, will provide distributed computing and debugging capabilities.

The LMS Optimus framework is currently being extended to provide parallel capability for scheduling DOE and NLP across networks of computers. A new version of DAKOTA has a capability for multilevel par-

allelism, in which several multiprocessor simulations are coordinated simultaneously.[33]

Because the frameworks examined do not yet provide the functionality necessary for implementing the complex problems required under HPCCP, development of a follow-on to FIDO[34] is proceeding. A major change in implementation is the use of a commercial CORBA-compliant system[35] instead of PVM to provide communications for distributed computation. CORBA is becoming accepted as the standard for distributed object technology. The wrapping of legacy code into Java modules (called "Java Beans") will promote flexibility in the construction of the problem by using Java visual programming packages. In addition, a commercial database will be used to promote data sharing among disciplines and provide access to persistent data. The redesigned framework will be used to implement an HSCT application that contains high-fidelity aerodynamics and structures codes, along with FLOPS[36] (Flight Optimization System) for the performance analysis.[37] This version of the HSCT problem increases the number of design variables to approximately two hundred.

Acknowledgments

The LaRC Multidisciplinary Optimization Branch would like to thank NASA LeRC representative Greg Follen (Computing and Interdisciplinary Systems Office) and Wright-Patterson Air Force Base representatives John Livingston (Aeronautical Systems Center), Max Blair and Jeff Zweber (both of Air Force Research Laboratory) for participating in discussions on frameworks and requirements. These discussions occurred during the evaluation period described in the "Introduction" section. The Branch would also like to thank Mark Hale (Georgia Institute of Technology) for his interaction with the branch during his graduate student work with IMAGE and for organizing a recent workshop on frameworks.[38]

References

[1] Sobieszczanski-Sobieski, J., and Haftka, R. T., "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," AIAA Paper 96-0711, Jan. 1996.

[2] Weston, R. P., Townsend, J. C., Eidson, T. M., and Gates, R. L., "A Distributed Computing Environment for Multidisciplinary Design," *5th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, AIAA-94-4372-CP, Sept. 1994, pp. 1091–1097.

[3] Web address <http://fmad-www.larc.nasa.gov/~ramakris/CAPSS/Papers/32G/review_frame.ps>

[4] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[5] Balling, R. J., and Sobieszczanski-Sobieski, J., "Optimization of Coupled Systems: A Critical Overview of Approaches," NASA CR-195019, also ICASE Report No. 94-100, Dec. 1994.

[6] Dennis, J. E., and Lewis, R. M., "Problem Formulations and Other Optimization Issues in Multidisciplinary Optimization," Center for Research on Parallel Computation, Rice University, CRPC-TR94469, Houston, TX, Apr. 1994.

[7] Tong, S. S., Powell, D., and Goel, S., "Integration of Artificial Intelligence and Numerical Optimization Techniques for the Design of Complex Aerospace Systems," AIAA Paper 92-1189, Feb. 1992.

[8] Guisset, P., and Tzannetakis, N., "Numerical Methods for Modeling and Optimization of Noise Emission Applications," *ASME Symposium in Acoustics and Noise Control Software, 1997 ASME International Mechanical Engineering Congress and Exposition*, Dallas, TX, Nov. 1997.

[9] Eldred, M. S., Hart, W. E., Bohnhoff, W. J., Romero, V. J., Hutchinson, S. A., and Salinger, A. G., "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, AIAA-96-4164-CP, Sept. 1996, pp. 1568–1582.

[10] Eldred, M. S., Outka, D. E., Bohnhoff, W. J., Witkowski, W. R., Romero, V. J., Ponslet, E. R., and Chen, K. S., "Optimization of Complex Mechanics Simulations with Object-Oriented Software Design," *Computer Modeling and Simulation in Engineering*, Vol. 1, No. 3, Aug. 1996, pp. 2406–2415.

[11] Evans, A., Lytle, J., Follen, G., and Lopez, I., "An Integrated Computing and Interdisciplinary Systems Approach to Aeropropulsion Simulation—NPSS," *International Gas Turbine and Aeroengine Congress and Exhibition*, Orlando, FL, June 1997.

[12] Ridlon, S., "A Software Framework for Enabling Multidisciplinary Analysis and Optimization," *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, AIAA-96-4133-CP, Sept. 1996, pp. 1280–1285.

[13] George, J., Peterson, J., and Southard, S., "Multidisciplinary Integrated Design Assistant for Spacecraft

(MIDAS)," *36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, New Orleans, LA, AIAA-95-1372-CP, Apr. 1995, pp. 1790–1799.

[14] Eidson, T. M., private communication, High Technology Corporation, Hampton, VA, May 1998.

[15] Zweber, J. V., Blair, M., Kamhawi, H., Bharatram, G., and Hartong, A., "Structural and Manufacturing Analysis of a Wing Using the Adaptive Modeling Language," *39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Long Beach, CA, AIAA-98-1758-CP, Apr. 1998, pp. 483–490.

[16] Hale, M. A., Craig, J. I., Mistree, F., and Schrage, D. P., "DREAMS and IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems," *Concurrent Engineering: Research and Applications*, Vol. 4, No. 2, June 1996, pp. 171–186.

[17] Web address <http://science.nas.nasa.gov/IT/ATED/>

[18] Walton, J. D., Korsmeyer, D. J., Batra, R. K., and Levy, Y., "The DARWIN Workspace Environment for Remote Access to Aeronautics Data," AIAA Paper 97-0667, Jan. 1997.

[19] Salas, A. O., and Rogers, J. L., "A Web-Based System for Monitoring and Controlling Multidisciplinary Design Projects," NASA TM-97-206287, Dec. 1997.

[20] Rogers, J. L., Salas, A. O., and Weston, R. P., "A Web-Based Monitoring System for Multidisciplinary Design Projects," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MI, AIAA-98-4706-CP, Sept. 1998.

[21] Grimshaw, A. S., and Wulf, W. A., "Legion—A View From 50,000 Feet," *Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing*, IEEE Computer Society, Los Alamitos, CA, Aug. 1996, pp. 89–99.

[22] Foster, I., and Kesselman, C., "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, Vol. 11, No. 2, 1997, pp. 115–128.

[23] Eidson, T. M., "A Programming Support Library for Distributed Memory Architectures," High Technology Corporation, HTC-9501, Hampton, VA, Mar. 1995.

[24] Breshears, C., "PVM Guide," Joint Institute for Computational Science, University of Tennessee, Jan. 1996, Web address <http://csep1.phy.ornl.gov/CSEP/PVM/PVM.html>

[25] Vanderplaats, G. N., "CONMIN—A Fortran Program for Constrained Function Minimization," NASA TM-62282, Aug. 1973, plus addendum, May 1978.

[26] Wrenn, G. A., "An Indirect Method for Numerical Optimization Using the Kreiselmeier-Steinhauser Function," NASA CR-4220, Mar. 1989.

[27] Welch, B., *Practical Programming in Tcl and Tk*, Prentice Hall, Upper Saddle River, NJ, 1995.

[28] Engineous Software, Inc., *iSIGHT Developer's Guide, Version 3.1*, Morrisville, NC, Apr. 1998.

[29] Engineous Software, Inc., *iSIGHT Advanced Designer's Guide, Version 3.1*, Morrisville, NC, Apr. 1998.

[30] Engineous Software, Inc., *iSIGHT Designer's Guide, Version 3.1*, Morrisville, NC, Apr. 1998.

[31] LMS Numerical Technologies, *LMS Optimus Revision 2.0 Users Manual*, Leuven, Belgium, Oct. 1997.

[32] Web address <http://sass577.endo.sandia.gov/9234/sd_optim/Dakota.pdf>

[33] Eldred, M. S., Hart, W. E., Bohnhoff, W. J., and Rhea, R. E., "Design and Implementation of Multilevel Parallel Optimization on the Intel TeraFLOPS," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MI, AIAA-98-4707-CP, Sept. 1998.

[34] Sistla, R., Dovi, A. R., and Su, P., "An Object Oriented Framework for HSCT Design," To appear in the proceedings of the *1998 NASA Computational Aerosciences Workshop*, Moffett Field, CA, Aug. 1998.

[35] Vinoski, S., "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," *IEEE Communications Magazine*, Vol. 14, No. 2, Feb. 1997, pp. 46–55.

[36] McCullers, L., "FLOPS—Flight Optimization System, Version 2.0, User's Guide," PRC Systems Services Report, Hampton, VA, Oct. 1986.

[37] Weston, R. P., Green, L. L., Salas, A. O., Samareh J. A., Townsend, J. C., and Walsh, J. L., "Engineering Overview of a Multidisciplinary HSCT Design Framework Using Medium-Fidelity Analysis Codes," To appear in the proceedings of the *1998 NASA Computational Aerosciences Workshop*, Moffett Field, CA, Aug. 1998.

[38] Web address <http://alpha.cad.gatech.edu/cfw>