NASA Technical Memorandum 100908

The Development of an Intelligent Interface to a Computational Fluid Dynamics Flow-Solver Code

(NASA-TM-100908) THE DEVELOPMENT OF AN INTELLIGENT INTERFACE TO A COMPUTATIONAL FLUID DYNAMICS FLOW-SOLVER CODE (NASA) 23 p CSCL 09B

N88-26127

Unclas
G3/61 0149971

Anthony D. Williams
Lewis Research Center
Cleveland, Ohio

Prepared for the
Symposium on Advances and Trends in Computational Structural Mechanics and Fluid Dynamics
cosponsored by George Washington University and Langley Research Center
Washington, D.C., October 17–19, 1988

NASA

THE DEVELOPMENT OF AN INTELLIGENT INTERFACE TO A COMPUTATIONAL FLUID DYNAMICS FLOW-SOLVER CODE

Anthony D. Williams
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

Researchers at the NASA Lewis Research Center are currently developing an "intelligent" interface to aid in the development and use of large, computational fluid dynamics flow-solver codes for studying the internal fluid behavior of aerospace propulsion systems. This paper discusses the requirements, design, and implementation of an intelligent interface to Proteus, a general purpose, 3-dimensional, Navier-Stokes flow solver. The interface is called PROTAIS to denote its introduction of artificial intelligence (AI) concepts to the Proteus code.

INTRODUCTION

Throughout the aeropropulsion community, researchers are striving to develop computer codes that will be able to compute the complex 3-dimensional fluid behavior internal to aerospace propulsion systems (refs. 1-4). The availability of such codes will enable the rapid assessment of new designs, significantly reducing the time, cost, and risk of developing future propulsion systems.

Typically, new computational fluid dynamics (CFD) codes are written for specific applications and evolve through many revisions and updates. Major revisions are usually required if a code is to be used for a different class of flow problems. Often, revising a code is extremely difficult because of poor documentation and a lack of a modular program structure. The result is that many CFD codes are not widely used. Thus, there is a great deal of "reinventing the wheel".

Many of the aforementioned problems are being addressed at the NASA Lewis Research Center in the development of a general-purpose Navier-Stokes flow solver called Proteus (ref. 5). This code incorporates a modular structure with disciplined programming and extensive documentation. The intent is to allow easy replacement and/or modification of Proteus modules to handle a broad class of flow problems.

While improved programming practices can do a lot to improve the usability and usefulness of CFD codes, even more dramatic improvements are possible through the use of "intelligent" interfaces to the codes. Such interfaces can provide guidance and assistance to users, automating many of the time consuming tasks associated with setting up and running a large CFD code. The interface can acquire and make available knowledge gained by previous users of the code. It's possible to have a user answer a few questions about the physics of the problem to be solved and to have the interface take care of setting up all of the input variables and creating and sending jobs to the appropriate super computer, workstation, etc.

Such an interface can be developed using artificial intelligence (AI) concepts (refs. 6-8). Artificial Intelligence is the science concerned with creating computers and computer programs that behave (or appear to behave) intelligently. Knowledge Based Systems (KBS) are AI programs that rely on encoded facts and heuristic knowledge rather than coded procedures to produce their results. Expert Systems are KBSs that perform complex tasks at the level of a human expert. These concepts are currently being applied in the business, medical, and engineering professions, as well as in academic and scientific research (refs. 9-11). This includes other efforts to create Knowledge Based Systems and Expert Systems for CFD-related applications (refs. 12-17).

Important reasons for choosing AI for the interface are: 1) the ease of representing facts, rules, and relationships using AI data structures; 2) the ability of AI systems to be easily expanded and modified; and 3) the abundance of software development tools currently on the market for designing and prototyping these systems. Combining these attributes with the productivity gains that are possible with today's AI development systems, allows multiple approaches to solving problems to be explored.

This paper discusses the requirements, design, and implementation of an intelligent interface to the Proteus code. The interface is called PROTAIS to denote the introduction of AI concepts to Proteus.

SYSTEM REQUIREMENTS AND SPECIFICATIONS

PROTAIS was initially envisioned as a way of providing inexperienced users of Proteus (and eventually other codes) with enough help and advice to study their problems with minimum supervision. As PROTAIS has evolved, it has acquired features that also benefit users who are familiar with a code's operation. Therefore, the PROTAIS design requirements have been expanded to accomodate both expert and novice users of a code. The following is a more detailed explanation of the current requirements and specifications for the PROTAIS System.

Functional Requirements

The primary function of the PROTAIS system is to automate the procedures required to set up and execute a user's problem on a given code. In the past, the user had to first learn the method for defining problems on the code. This could be as simple as modifying some of the input parameters, or as complex as rewriting sections of program code. The user was also required to learn the languages and protocols of the resident computer systems. Once these were understood, he could then create the input and control files necessary to run the problem.

For PROTAIS to automate this process, all of this information must be programmed into the system. It should also contain sufficient help and advice for its users. This allows users to quickly and easily set up and run their problems. The system should also be able to:

- Keep track of user problems and generate reports describing them.
- Obtain relevant information from a code's output file(s).
- Suggest the code setup for a problem, based on the results of similar problems stored in the KB.
- Predict the outcome of a test run based on previous runs and expert-provided rules.

The PROTAIS design should allow it to be easily modified for use on other CFD codes. This would enable the information acquired while using these codes to be stored in a common format that can later be combined and used for other more advanced applications. Then, as PROTAIS grows and advanced networking capabilities are added, researchers at different geographical locations could share information through the system. To help promote its widespread use, the design should also be generic enough to run with a variety of hardware.

Knowledge Base Requirements

The system's KBs are its principle means of storing and retrieving data. They must therefore be structured to best represent the relevant information from a flow solver's domain. This includes:

1. Detailed information about the code, its variables, and the interrelationships that exist between them.
2. Knowledge of the system's users, the computers networked to it, and their respective languages and protocols.
3. A history of all problems previously run on the system.
4. General facts and rules about CFD that may apply to the code or its

operation.

Item 1 specifies the requirements for defining a problem. It supplies the default values and allowable ranges for the code's variables, and the explanations that are displayed when a user asks the system for help. Information from item 2 controls user access to PROTAIS and the various computers used by the system. This is used primarily for submitting user jobs, and directing their output to the proper place.

Item 3 provides an easily accessible record of all problems that have been run on the system. This allows a researcher to easily keep track of his work. This information can also be used by others to help set up similar problems. Item 4 provides relevant CFD rules and principles that may help a user to better understand and define his problems.

Continued use of PROTAIS will inevitably create data storage problems in the future. One way of temporarily resolving this is to move the KBs onto a device with a large storage capacity like a mainframe computer. Doing this would also provide access to the many powerful data base management tools that are available for these systems, and facilitate multi-user access to the stored information.

Still, a strategy for limiting the size of the KBs must eventually be developed. This may entail deleting old, unused entries after a specific time; or perhaps, intelligently combining information from several problems in a manner that maintains the relevant information about them. Before developing this strategy, the relevant data to be stored for each problem must be identified. This is best done by observing the use of this data over a period of time. For this reason, only current KB requirements are being addressed now. The KBs will, however, be sufficiently generic to minimize any conversions that may be required later.

User Interface Requirements

A good user interface is essential to any system if it is to be used productively and efficiently, because all operations must be accessed through it. A good interface should simplify a user's tasks, and provide him with capabilities currently unavailable to him. It should also allow the user to customize certain aspects of its operation (eg., prompts, modes, etc.) to fit his individual preferences, and to access more advanced features as his experience and requirements grow.

The user interface of the PROTAIS System should optimize the operations associated with specifying and submitting user jobs, and allow a user to:

1. Easily view his work in a logical and consistent format.
2. Quickly identify and select subsequent command options.
3. Obtain help or assistance as needed at any stage in the program's execution.

The interface must accomodate users of different skill and experience levels, and be simple enough to use that anyone can run his problems with a minimum of training or outside help. Its display should allow a user to easily relate what he is doing on the screen, with what actually takes place in the code. Doing so automatically increases a new user's understanding of the code. It also makes the system easier to use by those who are already familiar with the code's operation.

SYSTEM DESIGN

The PROTAIS design consists of three levels and two user modes (fig. 1). The system was divided into 3 levels so that the various components could be used and tested during the development cycle. Each level adds more knowledge and capability to the system. Feedback obtained from the use of each level will contribute to the design of subsequent levels. This will help ensure that the final design meets the needs of all users.

Expert and novice modes will accomodate users based on their familiarity with a code's operation. All users are assumed to have a knowledge of fluid mechanics. Thus, an expert user is one who knows and understands the code and the computing environment in which it runs. A novice, on the other hand, generally understands his particular problem, but not how to use the code to study it.

The following is a more detailed description of how the various features will be introduced at each level.

Level 1

The first PROTAIS level comprises the basic system structure. It integrates the various code operating procedures into a single high-level interface. The KBs at this level will contain information about the system's users, the computers networked to it, the interfaced code and its default variable assignments, and the problems run on the system.

A highly-interactive user interface provides easy access to all system operations. Its multi-windowed display allows a user to quickly and easily define a problem and submit it for execution. These capabilities give the expert user the power and flexibility he needs to be most productive at his work. PROTAIS contains minimum intelligence at this level, so it can not provide the detailed help and advice that novice users would require. For this reason, a Level 1 Novice Interface has not been implemented.

Level 2

Level 2 adds to the system a well-defined knowledge of the relationships between the Proteus variables and their values. This information will be used to check data entered by a user before the code is executed. At this level, PROTAIS will be able to provide the user with detailed descriptions of the code's input variables and corresponding allowable values. It will also be able to provide examples of previously-run, similarly-structured problems to serve as models and detailed help at any time. Because of the availability of these additional services, both expert and novice interfaces will be implemented in level 2.

The Level 2 expert interface will be an enhanced version of Level 1, taking into account whatever feedback is obtained from previous users. The novice interface, on the other hand, will be totally different in appearance and form. It will be designed to shield the user as much as possible from the specifics of the code and the environment under which it is operating, while still providing him with the same capabilities of the expert user. Whereas the expert interface presents the user with a multifaceted display of information, the novice interface deals with much smaller blocks of information at a time.

Its interactions consist of a pseudo-dialogue between the user and the PROTAIS system. This allows the system to ask the user questions and to provide detailed explanations in a language and format that is familiar to him. Level 2 will also contain a high-level language or command set that will allow the user to describe his problems without having to think in terms of code specific variables and computer specific languages. The novice interface will train the user toward becoming an expert, thus allowing him to take advantage of the speed and efficiency provided by the expert interface.

Level 3

Level 3 is where most of the system's intelligence will be added. At this level, CFD principles applicable to the code and its use will be encoded in the system. This will be used to prevent anyone from entering a problem description that violates known physical laws.

The activities of the experts using the system will be studied. Then, an attempt will be made to capture, and store as rules, any techniques they use in their research to guide new or less experienced users. Inexperienced users will also be studied to identify and resolve problem areas in the novice interface.

Feedback obtained from both user groups will be used to improve the overall design.

Other features to be introduced at this level include a mechanism for processing the output files obtained while running a code. This could be as simple as checking a file to determine whether or not the code ran to completion or without errors. It could also be as complex as displaying the data on a high-resolution graphics screen or workstation, or searching the data for interesting characteristics.

Because of the time that will be required to explore and implement these enhancements, Level 3 is expected to evolve slowly. This, however, will enable PROTAIS to grow and to change with new generation flow-solver codes. As these codes emerge, they can be interfaced to PROTAIS and accessed through its main menu. The system could then advise users of the best code for their problems. This would not only provide users with a single common interface to these multifarious codes, but also allow for the exchange of information between them.

IMPLEMENTATION OF LEVEL 1

The first part of the PROTAIS System to be implemented was the Level 1 Expert Interface (grayed section of fig. 1). The following sections discuss the implementation of this level.

Development System

PROTAIS is being developed on a Symbolics 3600 series AI / Lisp (LISt Processing) Machine (ref. 18). The computing environment of this machine consists of a high-performance, custom-architecture, symbolic-processing-unit connected to a high-resolution graphics monitor that uses keyboard and mouse I/O. The machine's interactive programming environment consists of a large collection of system development and maintenance tools that support multiple programming paradigms. The machine is networked to other computers via an Ethernet connection using the TCP/IP protocol. This allows the transfer of jobs and files between PROTAIS and mainframe computers. The network also allows data to be sent to high-power graphics workstations where it may be displayed and analyzed.

The PROTAIS software is being written using a combination of Common LISP (refs. 19,20) and KEE (ref. 21) by IntelliCorp. LISP is a symbol manipulation language that has long been a standard for AI applications. For this reason, many sophisticated computing environments have been developed for writing and debugging large, complex programs in LISP. KEE (a Knowledge Engineering Environment) is a very powerful and dynamic software environment for developing KBS systems, that runs concurrently with the Common LISP language. It enhances the programming environment of the Symbolics machine, and provides the tools necessary for defining and accessing the knowledge bases and other data structures used by PROTAIS.

Knowledge Bases

The PROTAIS KBs use KEE data structures called units to represent objects within its domain. This domain covers all possible configurations of a code for defining problems. Each unit identifies a particular configuration or problem that has been run on the system, and contains information about the code, its variables, and the values assigned to them in that configuration. Each unit also contains an indication of how well the configuration ran, the names of similar or related units in the KB, the name of the user who created it, the date it was created, and any user-provided comments that describe the unit.

The PROTAIS units are grouped into three categories: masters, problems, and runs. Masters are used to distinguish between different versions of a code. Each master contains the default variable assignments used for all variables that the user does not explicitly define.

Problem units contain the information required to describe and run user problems. These units are usually attached to the master that represents the desired version of the code to be used (fig. 2(a)), so that they inherit its default variable assignments. This enables the user to define only those variables which require specific values (fig. 2(b)).

Runs are used to produce and submit computer executable jobs for a problem. Usually, a number of runs are generated for each problem to study the effects of making small changes to its variables (fig. 2(c)). Because of the inheritance of variable assignments and other characteristics that goes on between them, these units are often referred to as parents and children of each other. For example, in figure 2(c), the object labelled MASTER.SET can be viewed as the parent of PROBLEM.SET; which would conversely be considered its child. The same holds true for PROBLEM.SET and its children: RUN1, RUN2, and RUN3.

User Interface

Figure 3 shows the hierarchical display of the masters, problems, and runs generated in the system's KB Display Window. This display allows users to quickly and easily identify related items, and to access the information stored about them. The window can be scrolled to search through large knowledge bases, or pruned to display only the items that the user is working on at the time.

Each unit displayed has associated with it an option menu that lists all of the operations that can be performed on it (fig. 4). Selecting the "Display Item" option of this menu, opens another PROTAIS window that displays information stored about a selected unit (fig. 5). This window can be used to examine any units stored in the KB. It can also be used to modify or to add new information to the system.

To represent a new problem on the system, a unit must be created whose variable settings define the problem. This can be done by creating a problem unit as the child of some master (fig. 6(a)). This will usually entail redefining a significant number of variables, since a master usually represents the most general case. If the new problem were similar to one that had already been run on the system, then it could be created as a child of that problem unit instead, and thus, inherit the changes that were made to it (fig. 6(b)). The new unit would then already contain some of the desired characteristics. This would significantly reduce the amount of work required to define it.

Once a problem has been defined, the code can be set up and executed. This requires that a job control file be created that contains: 1) the commands required for the host to obtain the code, assemble the job, and execute it; 2) the input parameters required by the code to define the problem; and 3) any other code or system specific information required. An example of the computer JCL file that the system produces for a Proteus run is shown in figure 7. This file is automatically created and executed on the appropriate computer by a single PROTAIS user command.

PROTAIS is also capable of producing standard and customized reports that describe the problems stored in its KBs. Figure 8 illustrates a standard report that shows a list of all runs generated for a given problem. It also lists the variables changed to define the problem, and their values for each of the submitted runs. The customized report facility allows a user to select which variables and runs to include in the report and to define new variables as functions of standard ones.

When completed, Level 1 will be used to support further Proteus code testing. Knowledge acquired from this usage shall help further develop not only the Proteus code, but also Levels 2 and 3 of PROTAIS as well.

CONCLUDING REMARKS

This paper discusses the requirements and design of an intelligent interface to CFD codes, which novices and experts alike could use to increase their overall productivity. This interface will help inexperienced users learn to use new codes, and benefit from the knowledge previously acquired by others. Expert users will also benefit from a highly-interactive interface that automates the tedious and time consuming aspects of running a code.

Although, thus far, only the first level of the system has been implemented, benefits have already been realized. The Proteus developers have saved time in defining test problems, and were provided with a convenient method for visualizing and relating their problems. These benefits will multiply as the system expands to include more advanced capabilities.

REFERENCES

1. Shang, J. S.: An Assessment of Numerical Solutions of the Compressible Navier-Stokes Equations. AIAA Paper 84-1549, June 1984.
2. Anderson, Bernhard H.: Three-Dimensional Viscous Design Methodology of Supersonic Inlet Systems for Advanced Technology. AIAA Paper 84-0194, January 1984.
3. Benson, Thomas J.: Three-Dimensional Viscous Calculation of Flow in a Mach 5.0 Hypersonic Inlet. AIAA Paper 86-1461, June 1986.
4. Schwab, John R.; and Povinelli, Louis A.: Comparison of Secondary Flows Predicted by a Viscous Code and an Inviscid Code With Experimental Data for a Turning Duct. NASA TM 83575, February 1984.
5. Szuch, John R.: Application of Advanced Computational Technology to Propulsion CFD. NASA TM 100843, To be presented at the Symposium on Advances and Trends in Computational Structural Mechanics and Fluid

Dynamics, Washinton, D.C., October 17-19, 1988.

6. Nilsson, Nils J.: *Principles of Artificial Intelligence.* Tioga Publishing Co., 1980.
7. Winston, Patrick Henry: *Artificial Intelligence.* Second ed. Addison-Wesley Publishing Company, 1984.
8. Boden, Margaret A: *Artificial Intelligence and Natural Man.* Basic Books, Inc., Publishers, 1977.
9. Williamson, Mickey: AI techniques can raise productivity, reduce costs in various applications. *Computer Technology Review,* Volume VII, Number 4, April 1987, pp. 4-28.
10. Johnson, R. Colin: AI Works on Process Control. *Electronic Engineering Times,* August 3, 1987, p. 41.
11. Forbes, Mark: AI Expert Marvin Minsky -- Predicting the future of artificial intelligence. *Computer Technology Review,* Volume VII, Number 8, July 1987, pp. 1-25.
12. Arpasi, Dale J.; and Cole, Gary L.: Automating the Parallel Processing of Fluid and Structural Dynamics Calculations. NASA TM 89837, 1987.
13. Kutler, Paul; and Mehta, Unmeel: Computational Aerodynamics and Artificial Intelligence. AIAA Paper 84-1531, June 1984.
14. Tong, S. S.:Coupling Artificial Intelligence and Numerical Computation for Engineering Design. AIAA Paper 86-0242, January 1986.
15. Russo, Carol J.: Artificial Intelligence -- A new productive design tool. *The Leading Edge,* General Electric Company, Spring 1987, pp. 12-29.
16. Conner, R. S.; and Purdon, D. J.: PAN AIR Knowledge System. AIAA Paper 86-0239, January 1986.
17. Steinberg, S.; and Roache, P. J.: A Toolkit of Symbol Manipulation Programs for Variational Grid Generation. AIAA Paper 86-0241, January 1986.
18. Dukes, R.: On the Symbolics Artificial Intelligence Programming Environment.

AIAA Paper 86-0417, January 1986.

19. Steele, Guy L.: *Common LISP: The Language.* Digital Press (Digital Equipment Corporation), 1984.

20. Winston, Patrick Henry; and Horn, Berthold Klaus Paul: *LISP.* Second ed. Addison-Wesley Publishing Company, 1984.

21. Hedberg, Sara; and Stelzner, Marilyn: Knowledge Engineering Environment (KEE) System: Summary of Release 3.1. Intellicorp Technical Article, July 1987.

22. Fikes, Richard; and Kehler, Tom: The Role of Frame-Based Representation in Reasoning. *Communications of the ACM,* Volume 28, Number 9, September 1985, pp. 904-920.

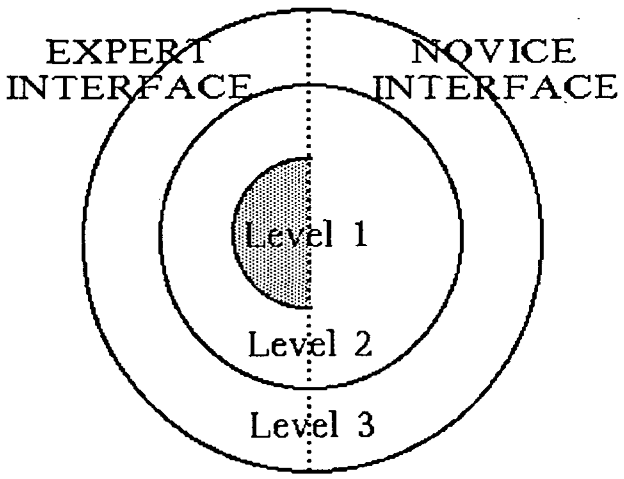
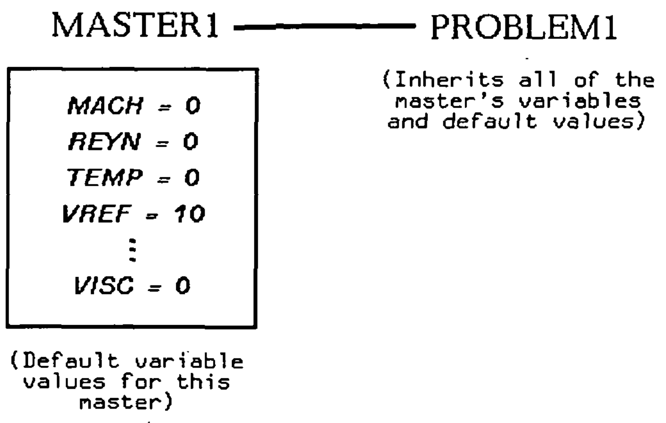
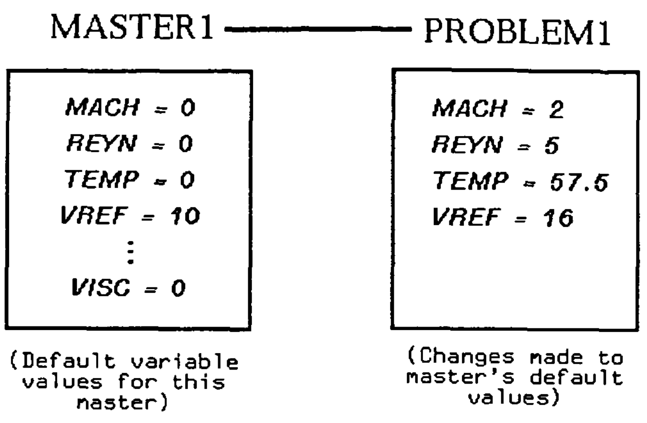


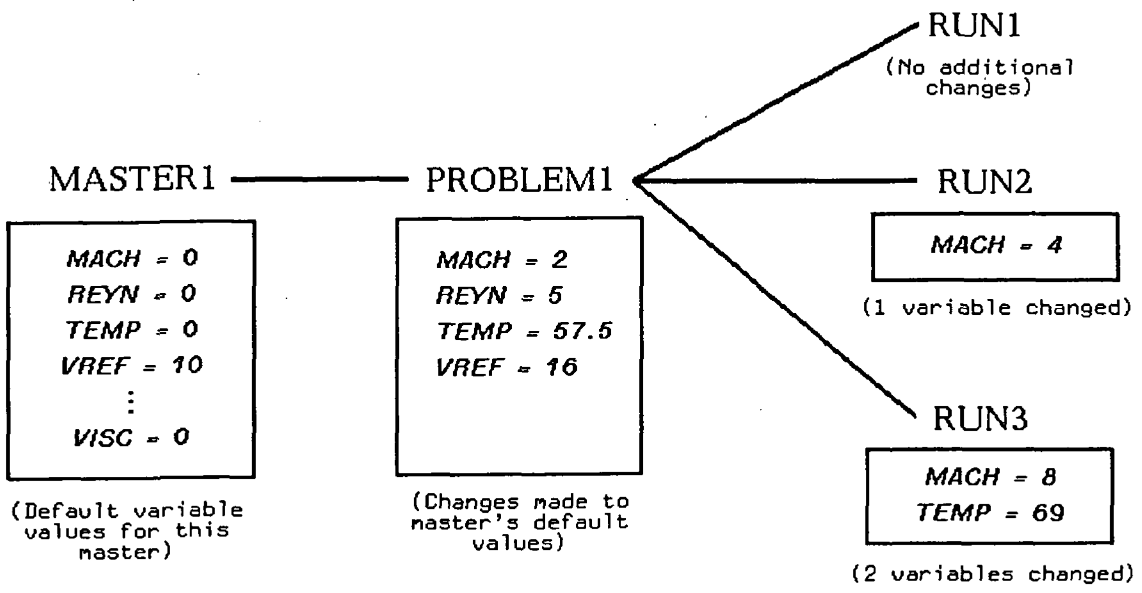
FIGURE 1. - IMPLEMENTATION DIAGRAM OF THE PROTAIS SYSTEM. (SHADED REGION REPRESENTS THAT PART OF THE SYSTEM CURRENTLY UNDER DEVELOPMENT).


MASTER1
PROBLEM1
(Inherits all of the master's variables and default values)
MACH = 0
REYN = 0
TEMP = 0
VREF = 10
⋮
VISC = 0
(Default variable values for this master)

2A - INHERITING DEFAULT VALUES FROM A MASTER.



2B - CHANGING A PROBLEM'S INHERITED VALUES.



2C - DEFINING RUNS FOR A PROBLEM.

FIGURE 2. - ILLUSTRATION OF THE INTERRELATIONSHIPS BETWEEN PROTAIS MASTERS, PROBLEMS, AND RUNS.

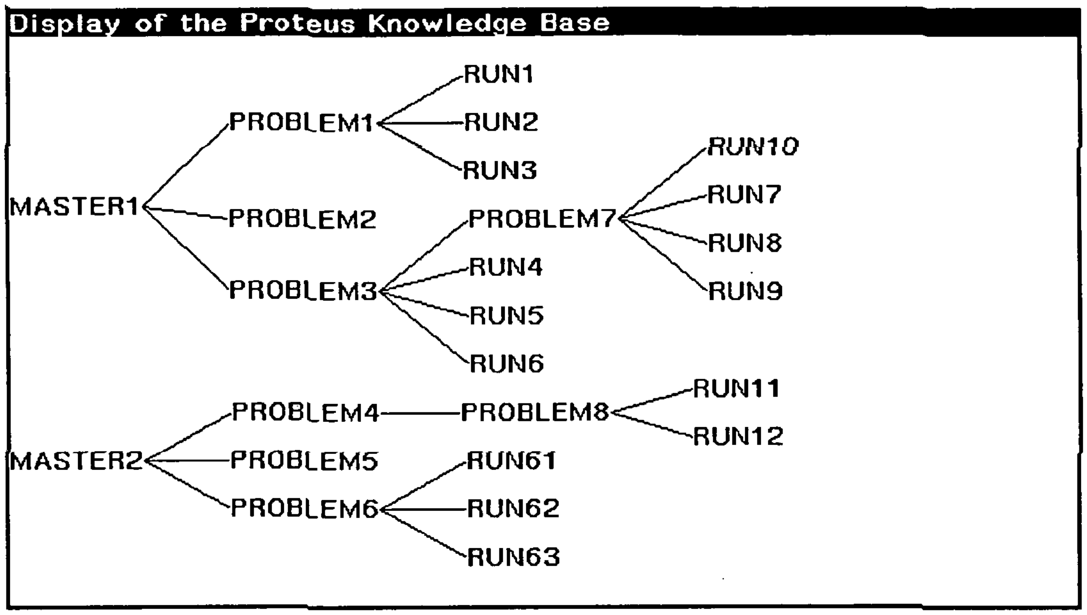


FIGURE 3. - SAMPLE DISPLAY OF UNITS IN THE PROTAIS KNOWLEDGE BASES. (AN ACTUAL DISPLAY CAN CONTAIN MORE RELEVANT NAMES FOR THE UNITS.)

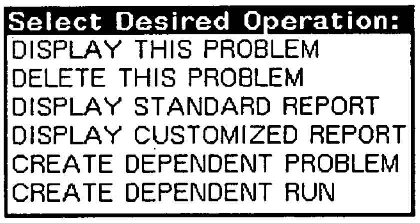
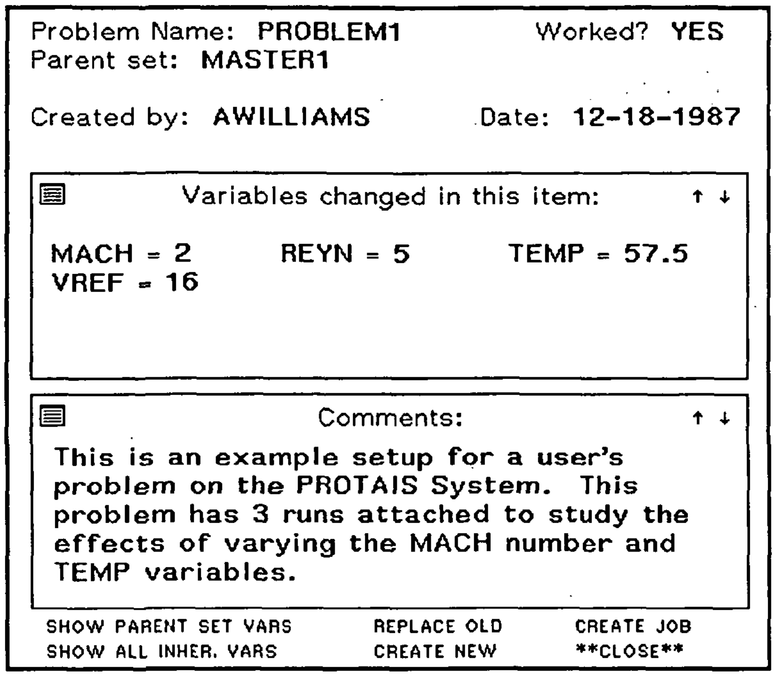


FIGURE 4. - OPTION MENU FOR A PROBLEM.


Problem Name: PROBLEM1 Worked? YES
Parent set: MASTER1

Created by: AWILLIAMS Date: 12-18-1987

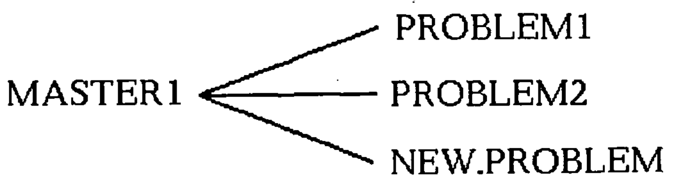
Variables changed in this item:

MACH = 2 REYN = 5 TEMP = 57.5
VREF = 16

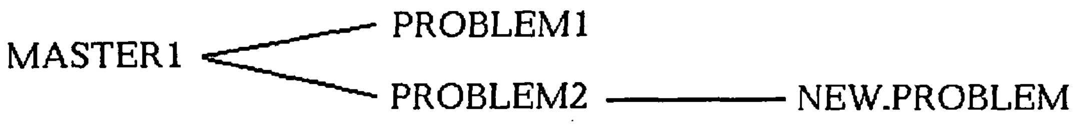
Comments:

This is an example setup for a user's problem on the PROTAIS System. This problem has 3 runs attached to study the effects of varying the MACH number and TEMP variables.

FIGURE 5. - PROTAIS WINDOW USED TO EXAMINE OR MODIFY SYSTEM INFORMATION.



6A - CREATING A NEW PROBLEM AS THE CHILD OF A MASTER.

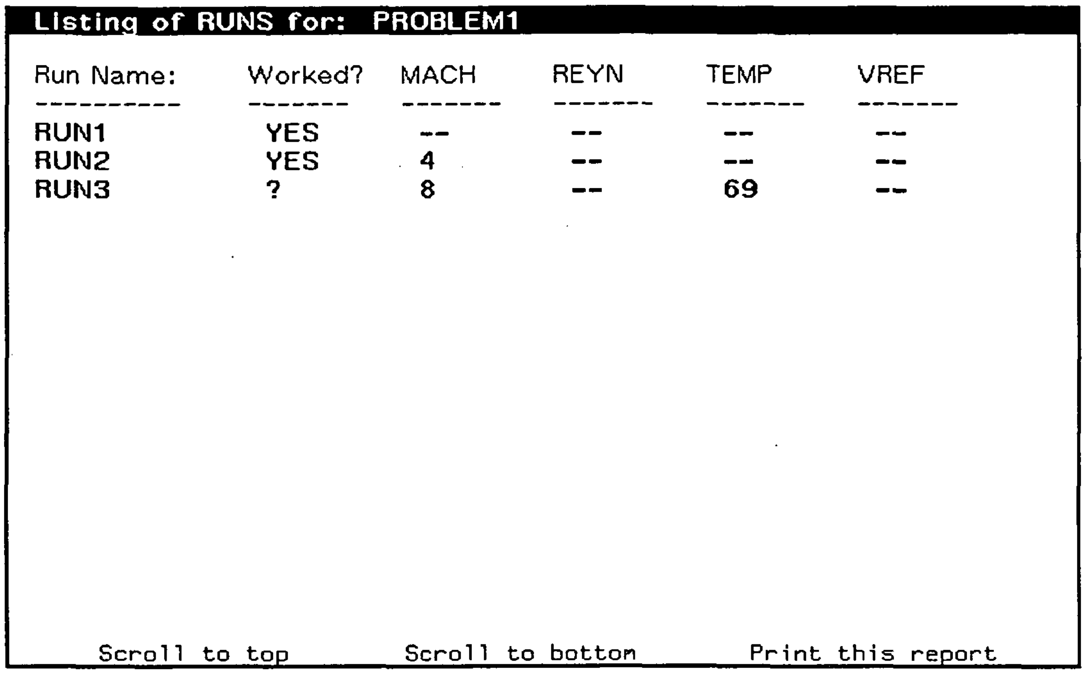


6B - CREATING A NEW PROBLEM AS THE CHILD OF A SIMILAR PROBLEM IN THE KNOWLEDGE BASE.

FIGURE 6. - PROCEDURES FOR ADDING NEW PROBLEMS TO THE PROTAIS KNOWLEDGE BASES.

```
JOB,JN=JOBNAME,MFL=500000,T=499.
ACCOUNT,AC=USERID,APW=USERPW.
ACCESS,DN=NPLOT,PDN=OUTPUT.DATA,ID=AWILLIAMS.
TASSIGN,DN=NPLOT,A=FT09.
ACCESS,DN=PROBJ,PDN=PROTEUS.CODE,ID=AWILLIAMS.
LDR,DN=PROBJ,L=0,NA.
SAVE,DN=NPLOT,PDN=OUTPUT.DATA,ID=AWILLIAMS,UQ.
DISPOSE,DN=NPLOT,DC=ST,MF=TS,DF=BB,TID=AWILLIAMS,NOWAIT.
EXIT.
DUMPJOB.
DEBUG.
SAVE,DN=NPLOT,PDN=OUTPUT.DATA,ID=AWILLIAMS,UQ.
/EOF


&RSTRT
&END

&FLAGS
&END

&GRID
&END

&REF
MACH
REYN
TEMP
VREF
&END

&BC
&END

&GMTRY
&END

&EXTRA
&END

/EOF
```

FIGURE 7. - PROTAIS-GENERATED JOB CONTROL LANGUAGE FILE REQUIRED TO RUN PROTEUS ON THE CRAY X-MP SUPERCOMPUTER.



Listing of RUNS for: PROBLEM1

```
Run Name:       Worked?    MACH      REYN      TEMP      VREF
----------      -------    -------   -------   -------   -------
RUN1             YES        --        --        --        --
RUN2             YES        4         --        --        --
RUN3             ?          8         --        69        --
```

FIGURE 8. - STANDARD REPORT GENERATED FOR PROTAIS PROBLEMS.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-100908	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle The Development of an Intelligent Interface to a Computational Fluid Dynamics Flow-Solver Code		5. Report Date
		6. Performing Organization Code
7. Author(s) Anthony D. Williams		8. Performing Organization Report No. E-4160
		10. Work Unit No. 505-62-21
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191		11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		13. Type of Report and Period Covered Technical Memorandum
		14. Sponsoring Agency Code

15. Supplementary Notes

Prepared for the Symposium on Advances and Trends in Computational Structural Mechanics and Fluid Dynamics, cosponsored by George Washington University and Langley Research Center, Washington, D.C., October 17-19, 1988.

16. Abstract

Researchers at the NASA Lewis Research Center are currently developing an "intelligent" interface to aid in the development and use of large, computational fluid dynamics flow-solver codes for studying the internal fluid behavior of aerospace propulsion systems. This paper discusses the requirements, design, and implementation of an intelligent interface to Proteus, a general purpose, 3-dimensional, Navier Stokes flow solver. The interface is called PROTAIS to denote its introduction of artificial intelligence (AI) concepts to the Proteus code.

17. Key Words (Suggested by Author(s)) AI Applications for CFD	18. Distribution Statement Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No of pages 22	22. Price* A02

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161