Constructing an Advanced Software Tool for Planetary Atmospheric Modeling

RICHARD M. KELLER, MICHAEL SIMS,
ESTER PODOLAK, AND CHRISTOPHER MCKAY

AI RESEARCH BRANCH, MAIL STOP 244–17
NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA–90–8–29–01

August, 1990

To appear in *Proc. 1990 International Symposium on Artificial Intelligence, Robotics and Automation in Space*

Constructing an Advanced Software Tool for Planetary Atmospheric Modeling

Richard M. Keller*, Michael H. Sims, Esther Podolak, and Christopher P. McKay*****

Information Sciences Division
NASA Ames Research Center
Mail Stop 244-17, Moffett Field, CA 94035 USA

Tel. (415) 604-3388 Fax. 604-6997
Keller@pluto.arc.nasa.gov (Internet) MHSims (NASAMail)

Abstract - Scientific model-building can be a time-intensive and painstaking process, often involving the development of large and complex computer programs. Despite the effort involved, scientific models cannot easily be distributed and shared with other scientists. In general, implemented scientific models are complex, idiosyncratic, and difficult for anyone but the original scientist/programmer to understand. We believe that advanced software techniques can facilitate both the model-building and model-sharing process. In this paper, we describe a prototype for a *scientific modeling software tool* that serves as an aid to the scientist in developing and using models. This tool includes an interactive intelligent graphical interface, a high-level domain-specific modeling language, a library of physics equations and experimental datasets, and a suite of data display facilities. Our prototype has been developed in the domain of planetary atmospheric modeling, and is being used to construct models of Titan's atmosphere.

1. Introduction and Motivation: Software support for scientific model-building

Model-building is an integral part of all scientific enterprise. Scientists studying a particular phenomenon develop theories in order to account for novel observations and to make predictions about expected behavior. To validate their theories, scientists conduct *in vivo* experiments whenever possible. Often, however, it is impracticable to carry out such direct experiments due to cost or other limiting factors. In these cases, scientists build models of the system under study and then test their theories against those models. Sometimes these models take the form of hardware (i.e., some sort of physical analog to the actual system), but often the models are expressed in software.

The construction of scientific software models can be a time-intensive and painstaking process. Many scientific models are written in terms of general-purpose numeric programming languages, such as FORTRAN, which have not been specially designed for the modeling task. Implementing a model can involve writing large and complex programs that access multiple datasets and utilize numerous different statistical and numeric processing packages. Development time for large scientific models may involve many months or years of effort.

For all the time and effort it takes to develop a model, the user community for most scientific models is limited to one -- the scientist who initially designed the model. This is not to say that one scientist has no use for another's models. On the contrary, model-sharing is highly desirable because it gives scientists the ability to "run experiments" and test their theories using different models without additional development overhead. The ability to easily inspect, use, and modify another scientist's models would be an extremely useful adjunct to current model-building practice, and an effective medium for communicating scientific ideas, as well. Just as scientists read technical papers describing theories, they should be able to inspect and exercise the software models that were used in validating those theories. Given the benefits to model sharing, why is it practiced so infrequently?

There are numerous technological barriers to scientific model sharing. Some of these barriers include:

- *Lack of comprehensibility*: Scientific software models are often sparsely commented and cryptic. Even a program that is initially well-written and commented will become fragmented over time.

---

* Employed under contract to Sterling Federal Systems.

** Employed under contract to Complex Systems Research.

*** Theoretical Studies Branch, Space Science Division, NASA Ames Research Center.

- *Wrong level of abstraction:* The model's structure is not obvious from the program code. The scientist is forced to interpret the code and translate low-level programming language constructs into high-level scientific terms.
- *Implicit assumptions:* Often important modeling assumptions are left implicit in the low-level code. These assumptions cannot be inspected or easily modified by new users.

Not surprisingly, some of these barriers are quite similar to those cited as discouraging traditional software sharing [1].

2. Objectives and Approach

Our primary research objective is to facilitate scientific model-construction and model-sharing by addressing the technological barriers described above. Despite the fundamental nature of scientific modeling, there is little software support available for this important activity. We are investigating the development of specialized software tools to ease the modeling process.

In particular, we believe that the following collection of advanced software techniques can substantially enhance the modeling process. We have begun to integrate these techniques in a *scientific modeling software tool* that serves as an aid to the scientist in developing and using models. The techniques include:

- *Interactive graphical interface*: To enhance comprehensibility and modifiability of models. Visual and iconic representations help the user to rapidly grasp the content of a model.
- *High-level modeling language*: To provide an appropriate level of abstraction for modeling and introduce natural domain concepts that are familiar to the scientist-user.
- *Analysis facilities*: To facilitate the interpretation of experimental results through the use of graphical plotting and statistical techniques.
- *Equation and dataset libraries*: To facilitate the sharing of standardized scientific equations and datasets.
- *Intelligent assistance*: To provide guidance and automate simple modeling steps. Artificial Intelligence-based techniques, such as constraint satisfaction, typed inheritance hierarchies, and backward-chaining control can reduce the amount of detail that the scientist-user needs to track.
- *Assumption maintenance facility*: To maintain explicit descriptions of modeling and data assumptions underlying a model and interdependencies among these assumptions.

Rather than attempting to construct a general-purpose modeling tool, our approach is to first focus narrowly on a particular scientific domain -- planetary atmospheric modeling -- and a specific class of models within that domain. Because the nature of scientific models differs widely across different scientific disciplines, we believe that software support can have the greatest impact by maintaining a narrow focus. Once we have some depth of experience in one scientific domain, we plan to widen and generalize our software techniques to accommodate other modeling problems within that domain and other similar domains. The next section describes the planetary atmospheric modeling domain.

3. Application to Planetary Atmospheric Modeling

The focal point for our preliminary research has been a model of the thermal and radiative structure of Titan's atmosphere. (Titan is one of Saturn's moons.) This model is implemented in a FORTRAN program (consisting of over 5000 lines of code) which was developed by co-author McKay [2]. The model was designed with two principal goals in mind: first, to be a general tool for investigating scientific issues associated with Titan, and second, to provide a mechanism for generating synthetic spacecraft data to be used in designing future instruments or in analyzing data from existing instruments. The McKay model is in considerable demand by planetary scientists -- both as a resource in support of ongoing Titan research and as an aid to study and planning activities associated with NASA's upcoming Cassini/Huygens Probe to Titan.

Our initial research has focused on a small portion of the overall Titan model -- the gas composition portion. The purpose of this sub-part of the model is to develop a profile of Titan's atmosphere that describes the pressure, temperature, and density of gases at various altitudes above its surface. This problem is underconstrained due to the shortage of empirical data on Titan. The major source of relevant experimental data is the Voyager I flyby of Titan back in November 1980. As Voyager I passed by the far side of Titan, it sent back radio

waves that passed through Titan's atmosphere and then on to receiving stations on Earth. Due to the gases in the atmosphere, the radio waves were refracted slightly as they passed through the atmosphere. The amount of refraction was measured at different altitudes above the surface. This refractivity data serves as a starting point for inducing the desired atmospheric profile in the Titan gas composition model [3].

In brief, the atmospheric profile is determined within the gas composition sub-model as follows (see Figure 1). First, for each atmospheric point to be profiled, the Voyager I refractivity data (R) is used to compute the number-density (ND) of the gases at that altitude. The number-density of a mixture of gases is defined as the number of molecules per volume of the mixture. Assuming the identity and relative percentages of gases in a mixture is known, the number-density can be computed as a function of refractivity. Next, using the molecular weight of the various gases in that mixture, the mass-density (RHO: mass per volume of mixture) can be computed from the number-density. The hydrostatic law can then be used to determine the pressure (P) from the mass-density by essentially summing the weight of the atmosphere above each elevation point in the profile. Finally, the temperature (T) can be determined from the mass-density and the pressure by applying an equation of state, such as the ideal gas equation. Figure 1 illustrates a level of abstraction at which a physicist might describe the problem, and is far more comprehensible than the corresponding FORTRAN code. Our goal is to construct a graphical interface and an associated modeling tool with which a physicist can construct a model with this level of abstraction.

4. Prototype

The aim of our prototype is to make the Titan model described in the previous section available for use by a larger group of planetary scientists. Currently, use of the Titan model is limited because the original programmer (co-author McKay) is the only one who can use and modify the FORTRAN model with any degree of facility. Our prototype permits scientists to inspect portions of the Titan gas composition sub-model, modify parts of the sub-model, execute the sub-model, and perform analyses on the results. Rather than augment the existing FORTRAN code, our approach has been to build a visual programming tool that enables a user to construct models graphically using a high-level atmospheric modeling language. The constructed model is automatically compiled into executable code by the system. The terms in the atmospheric modeling language involve domain concepts that are familiar to the scientist, including physical variables ("saturation point"), physics equations ("ideal gas law"), and experimental datasets ("Voyager refractivity data"). By interacting with a graphical interface, users essentially "program" using this more natural high-level modeling language.

For the purposes of our initial prototype, we have conceptualized model-building as a process of linking uncomputed physical variables to computed variables using *computational transformations*. For example, the process of linking the input Voyager I refractivity data to the output ideal gas temperature is accomplished by the sequence of transformations illustrated in Figure 1. Conceptually, each transformation takes as input a set of variables and produces a single variable as output. Physics equations and subroutines are two kinds of computational transformations supported in the current system. Physics forms the basis for atmospheric modeling, so the use of physics equations is natural in this context. The introduction of subroutines is motivated by the fact that in building models, scientists often makes use of program code that has been developed elsewhere. Examples range from standard numeric integration and data interpolation packages to complex scientific models developed by other scientists. The details of these imported

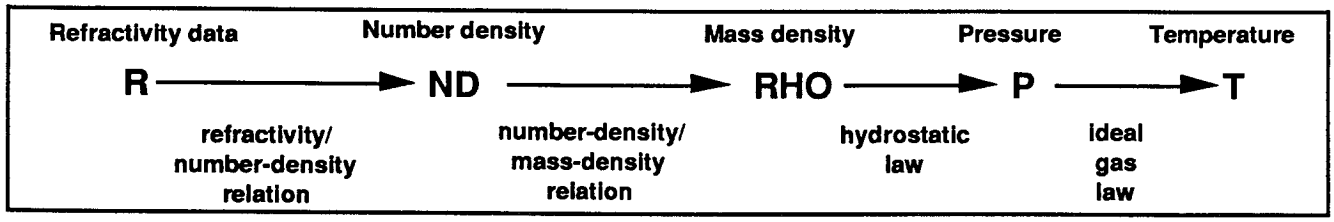

Figure 1: Determining the atmospheric profile

subroutines are assumed to be outside the scope and interest of the scientist-user, but can be incorporated into their models as 'black boxes'.

In our prototype, the process of linking variables using transformations is accomplished via a simple backchaining procedure. In this backchaining process, the user first selects a target physical variable they wish to calculate. Then the system presents the user with a set of transformations that can be used to compute the desired variable. The user selects one of these transformations, and the system checks to see whether all the input variables required by this transformation have already been calculated. If so, the transformation fires and the desired output variable is computed; if not, the backchaining process recurses for each of the uncomputed variables. During this process, the graphical interface displays a dependency-tree visualization of the current model as it is being built. This visualization is similiar to Figure 1, but in general depicts a more complex network of variables and transformations. The history of user modeling steps is recorded and maintained by the system, and can be displayed at any time.

The prototype features a Macintosh-like interface with pull-down menus and windows. The interface enables the user to visualize the model and its associated variables in a variety of formats, including plotted graphs and data tables for displaying computed variables. The interface provides various functions to manage the models stored in a scientist's workspace. For example, the user can switch the current focus of the model-building activity to a different model and/or workspace, and can retrieve old models, initiate new models, or delete existing models. Also, the interface provides a facility for applying user-defined tests of model viability to the current model under development. For atmospheric modeling, one such test is a test for atmospheric stability. If the temperature gradient predicted by a model is too steep, the atmosphere will be inherently unstable -- and this violates normal expectations. So the model testing facility provides the scientist with valuable feedback on whether the current model needs further refinement.

5. Status

To date, we have focused on model construction rather than model modification, although the tool supports simple types of changes to existing models. The modeling tool can be used to construct three variants of the basic gas composition model described in Section 3. This prototype has been implemented in LISP using Intellicorp's object-oriented knowledge representation tool KEE running on a Symbolics workstation. This development environment currently meets our needs for rapid prototyping. We envision eventual movement to a more suitable delivery platform, such as a Macintosh.

An atmospheric scientist brings a variety of different kinds of knowledge to bear in approaching the modeling task. This includes both domain knowledge, such as knowledge about physical variables, physics equations, and the structure of atmospheres, and modeling knowledge, such as various strategies and heuristics for atmospheric modeling, and various constraints and assumptions pertaining to the modeling process. We believe that representing knowledge of this type is the key to developing a useful modeling assistant to a human scientist. As yet, we have only represented knowledge about physical variables and physics equations in our prototype. We are currently focusing on representing domain objects (e.g., 'gas-parcels', 'gases', 'atmosphere-layers') and their interrelationships, and on representing scientific modeling assumptions. Our intent is to develop the current prototype into sophisticated modeling tool that enables active experimentation and sharing of models by atmospheric scientists. Further details on the current prototype and on our future plans can be found in [4].

6. References

[1] T.E. Cheatham, Jr., "Reusability Through Program Transformations", *IEEE Transactions on Software Engineering*, 10:5, pp. 589-594, September 1984.

[2] C.P. McKay, J.B. Pollack, and R. Courtin, "The Thermal Structure of Titan's Atmosphere", *Icarus*, **80**, 23-53, 1989.

[3] G.F. Lindal, G.E. Wood, H.B. Hotz, D.N. Sweetnam, V.R. Eshleman, and G.L. Tyler, "The Atmosphere of Titan: An analysis of the Voyager I radio occultation measurements", *Icarus*, **53**, 348-363, 1983.

[4] R.M. Keller, M.H. Sims, E. Podolak, C.P. McKay, and D.E. Thompson, "Proposal for Constructing an Advanced Software Tool for Planetary Atmospheric Modeling", Artificial Intelligence Research Branch Report #RIA-90-03-20-1, NASA Ames Research Center, March 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Dates attached	

4. TITLE AND SUBTITLE

Titles/Authors - Attached

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Code FIA - Artificial Intelligence Research Branch
Information Sciences Division

8. PERFORMING ORGANIZATION REPORT NUMBER

Attached

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Nasa/Ames Research Center

Moffett Field, CA. 94035-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Available for Public Distribution

Peter Friedland 5/14/92 BRANCH CHIEF

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstracts ATTACHED

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT