A Toolbox and a Record for Scientific Model Development

Thomas Ellman
Department of Computer Science
Hill Center for Mathematical Sciences
Rutgers University, New Brunswick, New Jersey 08903
ellman@cs.rutgers.edu

Abstract

Scientific computation can benefit from software tools that facilitate construction of computational models, control the application of models, and aid in revising models to handle new situations. Existing environments for scientific programming provide only limited means of handling these tasks. This paper describes a two pronged approach for handling these tasks: (1) designing a "Model Development Toolbox" that includes a basic set of model constructing operations; (2) designing a "Model Development Record" that is automatically generated during model construction. The record is subsequently exploited by tools that control the application of scientific models and revise models to handle new situations. Our two pronged approach is motivated by our belief that the model development toolbox and record should be highly interdependent. In particular, a suitable model development record can be constructed only when models are developed using a well defined set of operations. We expect this research to facilitate rapid development of new scientific computational models, to help ensure appropriate use of such models and to facilitate sharing of such models among working computational scientists. We are testing this approach by extending SIGMA, an existing knowledge-based scientific software design tool.

1 Problem: Support for Construction, Testing, Application and Revision of Scientific Models

Computational science presents a host of challenges for the field of knowledge-based software design. Scientific computation models are difficult to construct. Models constructed by one scientist are easily mis-applied by other scientists to problems for which they are not well-suited. Finally, models constructed by one scientist are difficult for others to modify or extend to handle new types of problems. Existing knowledge-based scientific software design tools, such as SIGMA [Keller and Rimon, 1992], provide only limited means of overcoming these difficulties. For example, SIGMA facilitates model construction by providing scientists with high-level data-flow language for expressing models in domain-specific terms. Although

SIGMA represents an advance over conventional methods of scientific programming, it supports only certain aspects of the model development process. In particular, SIGMA focuses mainly on automating the process of assembling equations and compiling them into an executable program. Construction of scientific models actually involves much more than the mechanics of building a single computational model. In the course of developing a model, a scientist will often test a candidate model against experimental data or against a priori expectations. Test results often lead to revisions of the model and a consequent need for additional testing. During a single model development session, a scientist typically examines a whole series of alternative models, each using different simplifying assumptions or modeling techniques. A useful scientific software design tool must support these aspects of the model development process as well. In particular, it should propose and carry out tests of candidate models. It should analyze test results and identify models and parts of models that must be changed. It should determine what types of changes can potentially cure a given negative test result. It should organize candidate models, test data and test results into a coherent record of the development process. Finally, it should exploit the development record for two purposes: (1) automatically determining the applicability of a scientific model to a given problem; (2) supporting revision of a scientific model to handle a new type of problem. Existing knowledge-based software design tools must be extended in order to provide these facilities.

2 Solution: A Model Development Toolbox and Record

We plan to attack this problem using two related ideas: First, we will define a "Model Development Toolbox". The toolbox will define a set of generic model development steps that are taken by most scientists in the course of developing scientific computational models. The envisioned generic steps include: (1) mapping equations onto physical situations; (2) fitting models against experimental data; (3) sanity checking model outputs against a priori sign, monotonicity or order of magnitude expectations; (4) testing models against experimental data; (5) analysis of test results; and (6) modification of models in response to test results. We plan to implement this toolbox in a scientific model development environment that guides scientist-users through the model development process. Second, we plan to design a "Model Development Record". The record will contain machine readable documentation of the entire model development process. To begin with, the record should describe the goals the model is intended to fulfill. For example, this might include a representation of the questions the model is (and is not) intended to answer The record should also describe the sequence of candidate models that were constructed in the course of developing the final model. For each candidate model, the record should describe: (1) the model itself; (i.e., equations and dataflow graphs), (2) assumptions underlying the model; (3) fitting techniques used to instantiate free parameters of the model; (4) sanity checks that were performed; and (5) tests against empirical data that were performed. The record should also describe (6) the temporal sequence of candidate models as well as (7) logical dependencies between test results on early models and modeling choices made in constructing subsequent, more refined models.

Tools for checking applicability of scientific models to new problems will rely heavily on the model development record. Important applicability checks include: determining

whether a proposed use of a model is consistent with the goals the model was originally intended to fulfill; determining if a new problem lies within the range of input parameter values for which the model was tested; and testing assumptions underlying the equations that were incorporated into the model. Each of these checks requires access to various aspects of the model development record. Likewise, tools that support model revision will also rely heavily on the model development record. Important types of model revision include: extending/modifying the model to handle a wider/different range of input parameters; re-fitting free parameters of the model to new empirical data; changing the assumptions used to model a physical process; adding/deleting physical processes to/from the model; and changing the overall purpose of the model. A model revision tool should automatically determine when a revision is needed (e.g., by determining that a new problem falls outside the range of problems handled by the original model, or by detecting discrepancies between empirical data and outputs of the model). It should suggest changes to the model that have the potential to cure the problem (e.g., by reasoning about sensitivities of outputs with respect to changes in intermediate results, or by reasoning about the effects of potential changes in assumptions on the outputs of the model). Finally the system should assist in re-validating the new model, (e.g., by suggesting new tests of validity, and carrying out and evaluating such tests.) In many cases, models may be revised by "replaying" a portion of the development record that led to the original model. Replay will require access to logical dependencies among test results and modeling choices found in the development record, using techniques similar to derivational analogy [Mostow, 1989] and transformational implementation [Balzer, 1985].

3 Model Development System Architecture

The overall architecture of our envisioned system is shown in Figure 1. The model development toolbox will serve as a front end to the whole system. The toolbox can interact with a human user to build an initial model in some scientific domain. It can also interact with a user in order to revise an existing model to handle a new situation. Finally, the toolbox also includes facilities for controlling the application of scientific models. As the toolbox guides the user through a series of model building, testing and revision steps, it interacts with several data bases. The model fragment data base contains the basic building blocks of scientific models. The toolbox uses techniques embodied in the SIGMA system to combine model fragments into one or more "current working models". As working models are constructed, they are tested against test data drawn from a test data base. Likewise, as tests are run, results are incorporated back into the test data base. As the initial model development process unfolds, the toolbox leaves a structured trace of the process in the model development record. When operating in replay mode, the toolbox is guided by a model development record constructed previously. Some portions of our system have already been implemented in SIGMA: These include the model fragment data base, the test data base and a framework for representing working models. Nevertheless, we expect that the representations used in SIGMA for these modules will need to be enhanced. A rudimentary version of the toolbox has also been implemented in SIGMA; however, most of our toolbox remains to be designed and build. The model development record is entirely new.

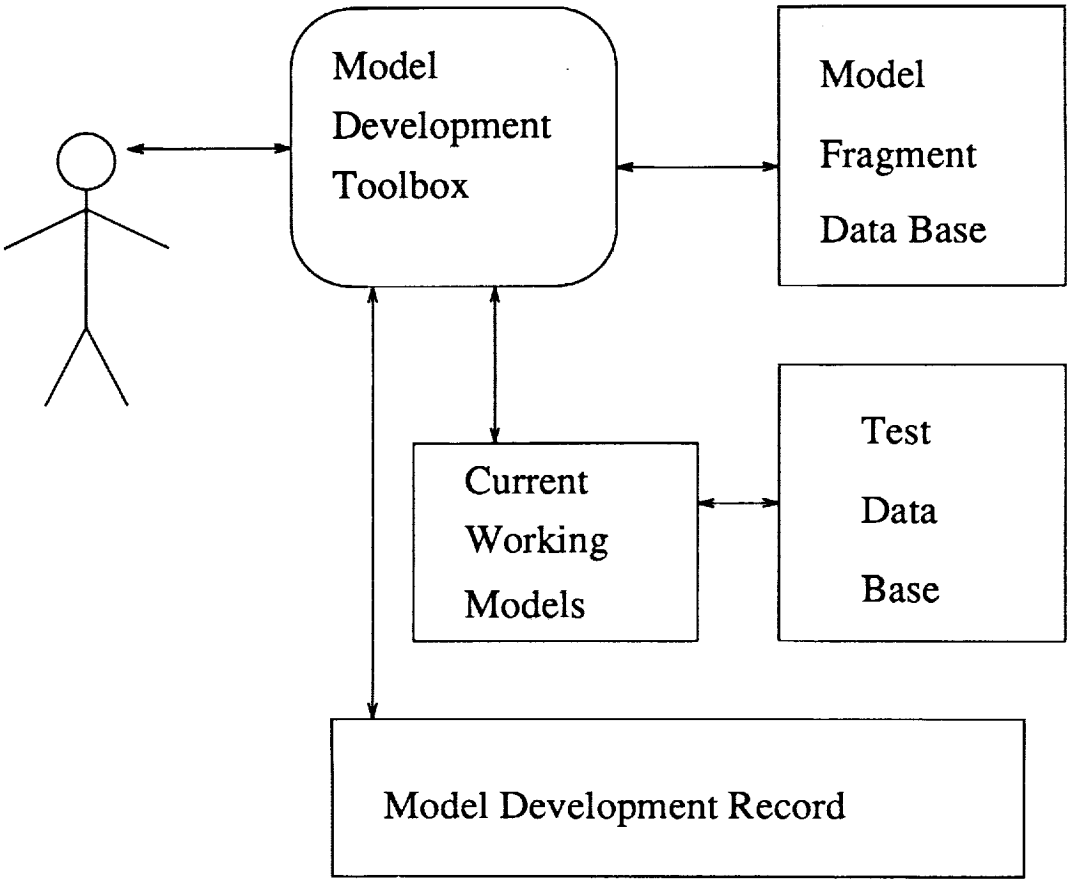

Figure 1: Model Development System Architecture

4 An Illustrative Example

As an illustration of the envisioned system, consider the following example of building a scientific model of the atmosphere of Saturn's moon Titan[1] The model takes as input a set of measurements of the refractivity of the atmosphere at various altitudes. The model is intended to compute atmospheric temperature and pressure at these altitudes. As the toolbox guides the human scientist through the model building process, it presents him with various modeling choices. For example he must decide which gases are to be included in the model. Let's suppose he chooses to include methane and nitrogen. He must also choose whether to use the ideal gas law, or a non-ideal gas law, to compute temperature from density and pressure. Let's suppose he chooses to use the ideal gas law. As the model is built, the user might declare certain expected properties of the output, e.g., that temperature and pressure are both positive numbers and are monotonically decreasing functions of altitude. The toolbox records these expectations in the model description in a representation that allows them to be checked automatically.

Once a preliminary model is constructed, the user may test the model on any available test

[1]The example is taken from [Keller and Rimon, 1992] and slightly modified. The details of example are not intended to be entirely accurate from the standpoint of atmospheric modeling.

data sets. If only input test data is available, (i.e., refractivity measurements) the system simply verifies that the outputs conform to declared expectations (i.e., the temperature and pressure are monotonically decreasing positive functions). If previously known output data is available, the system compares the known data to the outputs of the model and informs the user of discrepancies. For example, such tests might indicate that the pressure predictions are two low. The system might then suggest that the low pressure problem can be cured by either a change in the identities of the component gases, or by an addition of new gases into the mixture. Let's suppose the user decides to add ammonia into the mixture of gases. The system would revise the original model to include ammonia. It would also store the old model in the development record, along with a summary of the successful and unsuccessful tests performed on it. The cycle of model construction, testing and revision might be repeated several times before the user decides the model is satisfactory. The resulting model development record would include a description of the final model along with all the models examined along the way.

Once a satisfactory model is constructed by a human scientist, the model might be borrowed by a scientist working on a related problem, e.g., someone modeling the atmosphere of another satellite. The toolbox would guide such a new user through a series of steps designed to modify and validate the model for the new application. The system would examine the original model development record to determine what tests were performed on the original model. It would attempt to carry out analogous tests in the new setting. For example, the system might determine that, in the new setting, the model generates temperature or pressure levels for which the ideal gas law is not valid. The system would inform the user of the problem and suggest possible changes, e.g., using a non-ideal gas law, or changing the identities of gases in the mixture. Once the user chooses among the suggested revisions, the system would modify the model, update the record, and repeat any previous tests whose results are no longer valid. The cycle would repeat until the model passes all the tests suggested by the system and the user.

5 Key Research Issues

5.1 Model Development Toolbox Issues

A number of important research issues must be addressed along the way to implementing the model development architecture described in Figure 1. Implementation of the model development toolbox requires identifying a set of generic model building steps, and constraining the flow of control among them. Furthermore, in order that the toolbox support revision of scientific models, a number of distinct inference tasks must be performed. We thus expect to address the following questions in the course of designing the model development toolbox:

- *What primitive operations appear during the course of model development and model revision?* Potential primitives include: Select a model fragment to be used to compute a quantity. Replace one model fragment with another from the same class; Instantiate a generic model fragment in a specific scenario; Fit free parameters of a model against test data; Run a model on a set of test data; Compare test results to expected results; Add or remove a datum from the set of inputs or outputs of a model; Change the dimensionality of the inputs or outputs of a model.

- *What regularities appear in the sequences of operations that occur during model development and revision?* For example: Many models are hierarchically structured, i.e., they contain sub-models and sub-sub-models, etc. Potential construction strategies include: Top-down (breadth-first) and bottom-up (depth-first) or some combination. For each sub-model, the following sequence sequence of operations may be invoked: Select a model fragment incorporating suitable approximations; Run the model on a set of test data; Evaluate the test results; Revise the model fragment selection; Repeat, etc.

- *How can a system automatically detect circumstances in which a model must be revised?* For example: Input data can be compared to range constraints identified through previous tests; Output data can be checked for the expected sign, monotonicity or order of magnitude, when such expectations have been previously associated with the model; Outputs or intermediate results can be tested for consistency with simplifying assumptions; Outputs can be tested against benchmark data sets.

- *How can a system automatically determine which modeling choices must be revised to cure an identified problem?* A number of previously developed techniques may be applicable when suitably extended: For example, model selection methods that reason about the impact of choices on the sign of the error of a model's output are reported in [Addanki *et al.*, 1991] and [Weld, 1991]. Model selection methods that reason about the order of magnitude of the error may be developed by extending the techniques reported in [Raiman, 1991] and [Williams, 1991]. Likewise, model-selection methods relying on absolute error estimates may also be useful [Ellman *et al.*, 1993], [Falkenhainer, 1993] Furthermore, new techniques may be needed in order to reason about consistency between modeling choices in separate sub-models of a single larger model. Finally, truth-maintenance methods will likely prove useful in this portion of the system [De Kleer, 1986].

5.2 Model Development Record Issues

In order to design a model development record, we must identify the types of information that need to be included in the record, as well as suitable means of representing and organizing such information. The content of the record must be determined largely by the requirements of the processes the record is intended to support, i.e., developing models, controlling applicability of models and revising models. We thus expect to address the following questions in the course of designing the model development record:

- *What information about the goals of a scientific model must be represented in order to support development, application and revision of scientific models?* Potentially relevant information includes: A representation of the questions the model is intended to answer; A description of the quantities or relationships the models is (and is not) designed to compute; Desired accuracy levels; Legitimate and illegitimate uses of the outputs of the model.

- *What information about individual models and model fragments should be represented?* Aside from the models themselves, potentially relevant information includes: Restric-

tions on the input data; Testable simplifying assumptions that justify the approximations used in the model; Expectations regarding the sign, monotonicity or order of magnitude of the outputs or intermediate results.

- *What information about tests and test data should be represented?* Potentially relevant information includes: The purpose of the test; The model and test data used; Analyses performed on the test output data; Indications of satisfied and unsatisfied expectations.

- *How should the whole model development record be organized?* The record should include both the sequence of operations that led to the final model, as well as the development paths that failed and resulted in backtracking to earlier decision points. Thus the record needs to represent both temporal and logical relationships between different parts of the record.

- *What types of logical relationships between different parts of the record should be recorded?* Potentially relevant data includes: Dependencies between modeling choices in different parts of the model; Dependencies between goals and tests; Dependencies between test results and subsequent decisions.

We are pursuing this research by building an extension to the SIGMA system [Keller and Rimon, 1992] currently being developed at NASA Ames. We plan to develop the system by rationally reconstructing the process of developing and revising one of the two scientific models already implemented in SIGMA: a model of the atmosphere of Titan [McKay *et al.*, 1989], or a model of forest ecosystem processes [Running and Coughlan, 1988]. Additional candidate testbed domains include racing yacht design and jet engine nozzle design, each of which we have used as testbed applications for our previous work in the area of artificial-intelligence and computer-aided design [Ellman *et al.*, 1993].

6 Summary

The model development toolbox and record is expected to support a variety of activities that occur in the course of developing scientific computation models. These activities include construction and testing of new models; controlled application of models to specific problems, and revision of models to handle new situations. The system is also expected to promote rapid development of new scientific computational models, more reliable use of scientific models among computational scientists; wider sharing of scientific models within communities of scientists; and deeper understanding among scientists of the assumptions and modeling techniques incorporated in the models they use.

7 Acknowledgements

The research presented in this document has benefited from discussions with Richard Keller, Saul Amarel, Haym Hirsh, Lou Steinberg, Andrew Gelsey, John Keane and Mark Schwabacher.

References

[Addanki *et al.*, 1991] S. Addanki, R. Cremonini, and J. Scot. Graphs of models. *Artificial Intelligence*, 50, 1991.

[Balzer, 1985] R. Balzer. A 15 year perspective on automatic programming. *IEEE Transactions on Software Engineering*, SE-11(11):1257–1268, November 1985.

[De Kleer, 1986] J. De Kleer. An assumption-based tms. *Artificial Intelligence*, 28:127 – 162, 1986.

[Ellman *et al.*, 1993] T. Ellman, J. Keane, and M. Schwabacher. Intelligent model selection for hillclimbing search in computer-aided design. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., 1993.

[Falkenhainer, 1993] B. Falkenhainer. Ideal physical systems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., 1993.

[Keller and Rimon, 1992] R. Keller and M. Rimon. A knowledge-based software development environment for scientific model-building. In *Proceedings of the Seventh Knowledge-Based Software-Engineering Conference*, Tysons Corner, VA, 1992.

[McKay *et al.*, 1989] C. McKay, J. Pollack, and R. Courtin. The thermal structure of titan's atomsphere. *Icarus*, 80:23 – 53, 1989.

[Mostow, 1989] J. Mostow. Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence*, 40:119 – 184, 1989.

[Raiman, 1991] O. Raiman. Order of magnitude reasoning. *Artificial Intelligence*, 50, 1991.

[Running and Coughlan, 1988] S. Running and J. Coughlan. A general model of forest ecosystem processes for regional applications. *Ecological Modeling*, 42:125 – 154, 1988.

[Weld, 1991] D. Weld. Reasoning about model accuracy. Technical Report 91-05-02, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 1991.

[Williams, 1991] B. Williams. A theory of interactions: Unifiying qualitative and quantitative algebraic reasoning. *Artificial Intelligence*, 50, 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1994	Conference Publication

4. TITLE AND SUBTITLE
Seventh Annual Workshop on Space Operations Applications and Research (SOAR '93) – Volumes I and II

5. FUNDING NUMBERS

6. AUTHOR(S)
Kumar Krishen, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lyndon B. Johnson Space Center
Houston, TX 77058

8. PERFORMING ORGANIZATION REPORT NUMBER
S-749

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546
U.S. Air Force, Washington, D.C. 23304

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CP 3240

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Available from the
NASA Center for Aerospace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090
Subject Category: 99

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document contains papers presented at the Space Operations, Applications and Research Symposium (SOAR) Symposium hosted by NASA/Johnson Space Center (JSC) on August 3-5, 1993, and held at JSC Gilruth Recreation Center. The symposium was cosponsored by NASA/JSC and U.S. Air Force Materiel Command. SOAR included NASA and USAF programmatic overviews, plenary session, panel discussions, panel sessions, and exhibits. It invited technical papers in support of U.S. Army, U.S. Navy, Department of Energy, NASA, and USAF programs in the following areas: robotics and telepresence, automation and intelligent systems, human factors, life support, and space maintenance and servicing. SOAR was concerned with Government-sponsored research and development relevant to aerospace operations. More than 100 technical papers, 17 exhibits, a plenary session, several panel discussions, and several keynote speeches were included in SOAR '93. These proceedings, along with comments and suggestions made by panelists and keynote speakers, will be used to assess progress made in joint USAF/NASA projects and activities to identify future collaborative/joint programs. SOAR '93 was the responsibility of the USAF NASA Space Technology Interdependency Group Operations Committee. Symposium proceedings include papers presented by experts from NASA, the USAF, USA, and USN, U.S. Department of Energy, universities, and industry.

14. SUBJECT TERMS
Navigation; machine perception and exploration; ground operations teams; space physiology; operations challenges; artificial intelligence; robotics and telepresence research challenges; enhanced environments; medical operations; psychophysiology

15. NUMBER OF PAGES
839

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited