American Control Conference, Session WP-10, 15 June 1988, Atlanta, GA © AACC

A DATA-BASE MANAGEMENT SCHEME FOR COMPUTER-AIDED CONTROL ENGINEERING

James H. Taylor
Control Systems Lab
GE Corporate R & D
Schenectady, New York 12345

Ko-Haw Nieh
Information Systems Operation
GE Corporate R & D
Schenectady, New York 12345

Peter A. Mroz
Jackson Lab
DuPont Chambers Works
Deepwater NJ 08023

Abstract

There has been substantial progress made in the past decade in the development of analysis and design software for computer-aided control engineering (CACE). Engineering data-base management (EDBM) for support of CACE has not received much attention until recently, however. As CACE environments become more comprehensive and more powerful, the need for keeping track of the models, simulations, analysis results, control system designs, and validation study results over the control system design cycle becomes more pressing and the lack of EDBM support becomes more of an impediment to effective controls engineering.

We have defined and implemented a CACE environment with EDBM as an integral part. The data base is organized in a hierarchical framework having the levels *Project*, *Sub-project*, *Model*, *Attribute*, and *Element*. The levels *Project* and *Sub-project* accommodate several control engineers working on a single project, with a project lead engineer having responsibility and control of the entire data base. Within a project, *Models* (plant models, control system models, etc.) are the main focus. Each model has two attributes, a *Description* and a *Result_set*. Fundamental properties plus component models (representations of a plant, compensator, sensor, etc.) comprise the elements of a Description; elements of a Result_set include any data generated with the model, such as a time-history, frequency response, etc.

One factor that complicates the CACE DBM problem is that system models used for simulation, analysis, and design activity usually evolve as a project progresses. It takes a great deal of discipline to keep track of which results have been obtained using various instances of the model without some automatic support in the CACE environment. In our EDBM system (EDBMS), each model in the hierarchy is maintained using a *version control scheme* so that results obtained with each instance of the model can be unambiguously associated with that instance, thus maintaining data-base integrity. Two additional complicating features of this CACE data base framework preclude the use of a simple tree structure for the hierarchy outlined above:

1. Linearization of a nonlinear system produces both a *Result* and a *Component* that can be analyzed and have a *Result_set* of its own; this relation is maintained by use of a *Reference*.
2. Maintaining a component that may be used in a number of models (e.g., the sub-system **Plant** may be used in the models **Plant_Alone** and **NL_Fdbk_Syst**) without a proliferation of copies requires a second mechanism we call a *Link*.

All of these situations are managed in the CACE data-base schema presented below.

1. INTRODUCTION

Software environments for computer-aided control engineering (CACE) have been in a stage of rapid development over the past decade. Primary emphasis has been placed on the following aspects:

- improving or extending core CACE capabilities, i.e., nonlinear simulation, identification, and linear analysis and design in the frequency and time domains;
- integrating core CACE functionality (e.g., nonlinear simulation with linear analysis and design);
- improved interactive user interfaces;
- addition of artificial intelligence support; and
- migration to engineering workstations and personal computers for hosting part or all of a CACE environment.

Given these developments, it is surprising that support for engineering data-base management has lagged so far behind. As CACE environments became more powerful in terms of systems analysis and design capabilities and broader in terms of the variety of CACE activity supported, the need for keeping track of the models, analysis results, control system designs, and validation study results has become more important. In a real industrial control system design project, the large number of files generated in the complete design cycle and the relations among these files can be very difficult to comprehend and manage by manual means. Controls

engineers who have been successful at managing their engineering data base should be congratulated, for the discipline required to manually maintain and document the files, listings, hard-copy plots, etc. that arise over the design cycle can be prodigious. It is often the case that, after the passage of a few months, the engineer cannot really say exactly how a given result was obtained and thus cannot reproduce it, to the consternation of all concerned.

These considerations have motivated us to define and implement a CACE environment with EDBM as an integral part. Preliminary requirements and concepts were outlined in [1] and an effective user interface for such a system is described in [2]; the complete system is described below.

2. *THE CACE DBM PROBLEM*

We indicated above that EDBM is important. The magnitude of the control engineer's EDBM problem can best be appreciated by looking at CACE in a larger context than linear analysis and design. In many cases, the user starts with a nonlinear model of the process to be controlled, and progresses through the following range of CACE activity: nonlinear simulation (e.g., model validation and behavioral analysis), trim or equilibrium determination (defining operating points), linearization, linearized system analysis and design, nonlinear control system design, and control system validation using both linear and nonlinear models of high fidelity [3, 4]. Much of this activity is exploratory and iterative in nature. Several systems have been developed that cover most or all of this gamut - Ctrl-C + ACSL, PRO-MATLAB + ACSL, and MATRIX$_x$ + SYSTEM_BUILD are well-known commercial examples†, and the GE Federated System [5] represents our efforts in this area; however, none of these manage the resulting DB beyond implementing rudimentary file-naming conventions. Even systems with ancillary expert systems to "expert aid" the CACE process (e.g., [6 - 11]) do not provide such support at the present time.

In many applications,, control engineering activity develops a substantial DB. In flight control, for example, a typical DB may contain one nonlinear airframe model, 20 linearized models (corresponding to 20 points in the flight envelope), 20 linear control system designs, one or several candidate nonlinear ("full-envelope") control systems, and innumerable time-histories and analysis results (step responses, equilibria, eigenvalues, frequency responses, root-locus data, singular values, . . .).

The above DB picture may be multiplied many times over by two additional factors: First, in multi-disciplinary applications such as integrated flight and propulsion control, one can produce an enormous DB by combining 20 flight regimes with 9 engine operating conditions, for example. Second, it is not unusual for the primary nonlinear model of the controlled object to be modified several times over the complete analysis and design cycle. We can thus 'size' the EDBM problem by assuming three instances of each model and perhaps six results per nonlinear or linear model on the average, giving us the final product $3 \times 6 \times (1 + 20 + 20 + 2 + 1 + 9 + 9 + 1) = 1134$ files to manage.

As the above cases make clear, CACE project activity can generate hundreds of results and model files, of which the user may wish to retain and manage a substantial percentage. This may not be a "large" DB in terms of commercial DBM systems, but it is neither small nor simple, and is thus difficult for most human users to manage effectively without support.

3. *CACE DATA BASE DEFINITION*

As mentioned previously, the CACE data base is organized in a hierarchical framework having the levels *Project*, *Sub-project*, *Model*, *Attribute*, and *Element*, as depicted in Fig. 1. This architecture represents a refinement of the scheme outlined in [1], and was developed from a careful analysis of the informal organization that is traditional in the field plus knowledge of CACE work-patterns and data element interrelations.

The top levels *Project* and *Sub-project* accommodate several control engineers working on a single project, with a project lead engineer having responsibility and control of the entire data base, particularly the primary shared models and results (e.g., plants and controller designs). Other engineers work in their own subset of the data base, using both shared and private DB elements.

All CACE activity within a project or sub-project is carried out in the context of *Models* (plant models, control system models, etc.). Each model has two attributes, a *Description*, i.e., a catalog of the fundamental characteristics of the model (linear or nonlinear, continuous- or discrete-time or mixed, etc.) plus a list of the sub-systems or components that comprise the model and a definition of the sub-system interconnections, and a *Result_set*, i.e., the set of simulation, analysis, and design results obtained using the model. *Elements* of a Description characterize the system in terms of component models, e.g., representations of a plant, compensator, sensor, etc.; elements of a Result_set

† Ctrl-C, ACSL, PRO-MATLAB, and MATRIX$_x$ + SYSTEM_BUILD are trademarks of Systems Control Technology Inc., Palo Alto CA; Mitchell & Gauthier Assoc. Inc., Concord MA; and Integrated Systems Inc., Santa Clara CA respectively.

include any data item generated with the model, such as a time-history, frequency response, etc. Note that we have not defined the data structures at the element level; in light of recent activity directed at establishing standards in CACE data structures [12 - 14], we have decided to accept the data structures and formatting produced by existing CACE packages and incorporate data element "filters" to transform from one format to another where required.

The most difficult aspect of CACE DBM is maintaining the integrity of the DB in situations where system models evolve over the term of the project as more accurate model information is obtained [1]. It is almost always true that models created early in a project have to be modified, either to reflect additional information (e.g., experimental results), changes in the design of the plant itself, or the discovery that the original model is not valid over the range of operation occurring in the validation and acceptance part of the design cycle. The loss of integrity in the sense of not knowing which analysis and design results were obtained with which instance of the model is the most common fault encountered in documenting the control system design cycle, and the outcome is irreproducible results, unsupportable design decisions, and unproductive re-iteration. The solution to this problem is to be found in the software engineering discipline of *version control.* In our system, each model in the hierarchy is maintained using a version control scheme such as found in the DEC product DECVAX/CMS† [15], so that system models can evolve and results obtained with each instance of the model can be correctly related with that instance. Specifically, each instance of the model is identified by a *name* and *class number:* <model_id> = <model_name> + <class_number>. The Description of each model instance is maintained in relational (tabular) form in terms of the *Version* for each *Component model;* this is illustrated in Fig. 2. Components are maintained in the form of file names pointing to the actual data elements that are maintained in CMS libraries. Any version of a component can be obtained using a CMS Fetch command; therefore, any model instance can be assembled for documentation or further analysis. Observe the existence of a *Notes* field to permit the association of engineering comments with each <model_id>; there may also be notes related to each component model, which are tabulated in the next level of the hierarchy.

† DEC VAX/CMS is a trademark of Digital Equipment Corp., Maynard MA.

Two additional complicating features of this CACE data base framework [1] have made it impossible to use a simple tree structure for the hierarchy outlined above:

1. Linearization of a nonlinear system produces both a *Result* and a *Component* that can be incorporated into a model, analyzed, and thus have a Result_set of its own; this relation is maintained by use of a *Reference*. Note that if the linear sub-system model is subsequently altered to produce a new version, then the Reference applies only to the original version.

2. Maintaining a component that may be used in a number of models (e.g., the sub-system **Plant** may be used in the models **Plant_Alone** and **NL_Fdbk_Syst**) without a proliferation of copies requires a second mechanism we call a *Link.* The idea of a data-base link is adopted from UNIX†; here, we mean that the same CMS Library file location of the component **Plant** is used in any Description of a model containing it. The Link mechanism ensures that sub-system models can be maintained with integrity.

Both of these situations are portrayed in the CACE data-base schema in Fig. 3.

The importance of the Reference feature can be illustrated by considering the flight control systems design scenario outlined in Section 2: Assume that **Plant** in Fig. 3 represents a nonlinear model of an aircraft's aerodynamic behavior. In order to achieve a "full-envelope" control system design, the standard practice is to define several dozen flight regimes for different altitudes and Mach numbers, linearize at each flight condition, perform a linear analysis and design at each point, and combine the set of linear designs via gain scheduling. In this scenario, there exist perhaps 20 linear models that are used to generate substantial Result_sets of their own; if it is not possible to trace each linear model back to its *Parent Result* (Result_set entry for a particular instance of the nonlinear plant model and corresponding to a particular set of conditions such as operating point), then documenting the design and even performing the gain scheduling part of the design may be impossible. In the general case, the reference mechanism is the key to relating linearized models and their associated results with their origin (<model_id> *and* operating regime) with integrity.

The Link concept is used to solve another integrity problem: model proliferation. It is difficult enough to

† UNIX is a trademark of AT&T.

maintain and track one model of a particular component as it is refined and edited; if several copies of the model exist and have to be maintained as separate entities by manual means, then the problem is seriously compounded. Our EDBMS solves this problem by maintaining only one copy of each sub-system model. Every time it is modified, a new version is created, as mentioned above; each model's <model_id> is tied to a specific version of each component. Note that each model using a linked component may or may not have an instance corresponding to each sub-system version, as illustrated in Fig. 4. While the component is associated with one model, the fact that there can be sub-system model versions that are unique to any other model provides no loss of generality. The main care that must be taken in treating linked component models relates to purging model classes and deleting entire models; for example, if the model "owning" a linked sub-system is deleted, that component must be transferred to another "owner" model.

The lowest level of the EDB hierarchy under *Description* is directly evident from Fig. 2. The components of a model exist and are managed in files that contain code appropriate to the modeling language(s) supported in the CACE software for simulation, analysis, and design. Therefore, we have not provided examples of these data elements. The elements under *Results_set* are less traditional in their organization, as shown in the result tabulation in Fig. 5. The result itself (whose name and type are indicated in columns 1 and 2) is clear enough; it may correspond to a file containing time-history data, eigenvalues, etc. The presence of notes associated with each result (indicated in the last column) is also straightforward. The item not described so far is the *Condition_Spec* entry in column 3 of the table. This is required to track the one remaining factor that governs the integrity (reproducibility, etc.) of the CACE DB, i.e., the conditions under which the result was obtained. This is especially important for nonlinear aspects of CACE, such as simulation, equilibrium determination, linearization, etc. where the user may specify arbitrary initial conditions, input amplitudes, and parameter values, and use various algorithms and specifications (e.g., tolerances and iteration limits). This information is maintained in a separate file that the EDBMS tracks and associates with one or more item in the Result_set, as illustrated.

More detail concerning the contents, structure, and interface of the CACE DB managed by the scheme presented here may be found in [2]. The interface to the EDBMS has not been discussed so far, since most of the motivation and design of the EDBMS is based on other considerations. It is important to note, however, that a well-designed interface can further enhance the effectiveness of the CACE environment by making unnecessary detail transparent to the user. Several examples of "information hiding" are included in Figs. 2 and 5, where it may be observed that names are assumed to be supplied by the user wherever possible (e.g., NL_Fdbk_Syst, Cruise_20k_Mp6), while information such as file physical location (disk, subdirectory, name) is not displayed unless requested. The file names would generally be based on file-naming conventions that create unique designations that may encode project, model_id, component_id, result_type etc. and thus (as is often true with user-supplied file names as well!) difficult to recall accurately. Also, the user only needs to know whether or not a note exists (Yes/No in the last column of Figs. 2, 5); the EDBM can locate it for display if the user requests this action.

4. SUMMARY AND CONCLUSIONS

We have demonstrated that rigorous engineering data-base management for computer-aided control engineering is both important and achievable. A hierarchical organization of CACE data base elements has been presented, and additional mechanisms for maintaining data-base integrity have been described. Many of these concepts were implemented and proven to be feasible in a rapid prototype of the final CACE environment; this effort is described in [16, 17]. Version 1.0 of the final environment will be tested in May 1988, and a fully operational system will be completed this summer.

A second aspect of this work has not been treated here: the integration of EDBM with the rest of the CACE environment. This is a critical consideration, because a poor interface will result in little or no user acceptance. This topic is considered in a companion paper [2]. It is hoped that these contributions will serve as the basis for more supportive CACE environments in the future.

Acknowledgement: The definition and development of the EDBM scheme described in this paper was sponsored in part by:

Flight Dynamics Laboratories
Air Force Wright Aeronautical Laboratories
Aeronautical Systems Division (AFSC)
United States Air Force
Wright-Patterson AFB, Ohio 45433-6553

REFERENCES

[1] Taylor, J. H., "Conventional and Expert-Aided Data-Base Management for Computer-Aided Control Engineering", *Proc. American Control Conference*, Minneapolis, MN, June 1987.

[2] Mroz, P. A., McKeehen, P., and Taylor, J. H., "An Interface for Computer-Aided Control Engineering Based on an Engineering Data-Base

Manager," *Proc. American Control Conference,* Atlanta, GA, June 1988.

[3] Taylor, J. H., "Environment and Methods for Robust Computer-Aided Control Systems Design for Nonlinear Plants", *Proc. Second IFAC Symp. CAD of Multivariable Technological Systems,* West Lafayette, Indiana, September 1982.

[4] Taylor, J. H., "Computer-Aided Control Engineering Environment for Nonlinear Systems", *Proc. Third IFAC Symp. CAD in Control and Engineering Systems,* Lyngby, Denmark, August 1985.

[5] Spang, H. A., III, "The Federated Computer-Aided Control Design System", *IEEE Proceedings, Vol. 72,* 1724-1731, December 1984.

[6] Taylor, J. H. and Frederick, D. K., "An Expert System Architecture for Computer-Aided Control Engineering", *IEEE Proceedings, Vol. 72,* 1795-1805, December 1984.

[7] Birdwell, J. D., J. R. B. Cockett, R. Heller, R. W. Rochelle, A. J. Laub, M. Athans, L. Hatfield, "Expert systems techniques in a computer-based analysis and design environment", *Proc. Third IFAC Symp. on CAD in Control and Engineering Systems,* Lyngby, Denmark, August, 1985.

[8] L. S. Su, O. Ahrif, G. L. Blankenship, "An Expert System Using Computer Algebra for the Analysis of Nonlinear Control Systems", *Proc. American Control Conference,* Seattle, WA June 1986.

[9] T. L. Trankle, P. Sheu, and U. H. Rabin, "Expert Systems Architecture for Control System Design", *Proc. American Control Conference,* Seattle, WA June 1986.

[10] G. K. H. Pang, "An Expert System for CAD of Multivariable Control Systems Using a Systematic Design Approach", *Proc. American Control Conference,* Minneapolis, MN, June 1987.

[11] J. H. Taylor, "Expert-Aided Environments for CAE of Control Systems", Plenary Lecture, *Proc. 4th IFAC Symp. on CAD in Control Systems '88,* Beijing, PR China, August 1988.

[12] Maciejowski, J. M., "Data Structures for Control System Design", *Proc. EUROCON '84,* Brighton, UK, 1984.

[13] Rimvall, M., "A Structural Approach to CACSD", in *Computer-Aided Control Systems Engineering,* M. Jamshidi and C. J. Herget, Editors, Elsevier Science Publishers, 1985.

[14] Maciejowski, J. M., "Data Structures and Software Tools for CAD of Control Systems: A Survey", *Proc. 4th IFAC Symp. on CAD in Control Systems '88,* Beijing, PR China, August 1988.

[15] *User's Introduction to VAX DEC/CMS,* Digital Equipment Corp., Maynard MA, November 1984.

[16] Nieh, K-H., and Mroz, P. A., "GNU-Emacs: An Excellent Environment for Software Prototyping", *9th GE Software Engineering Conference,* Daytona Beach, FL, May 1988.

[17] Mroz, P. A., "Rapid Prototype Software for Computer-Aided Control Engineering", Master of Science Thesis, Rensselaer Polytechnic Institute, November, 1987.

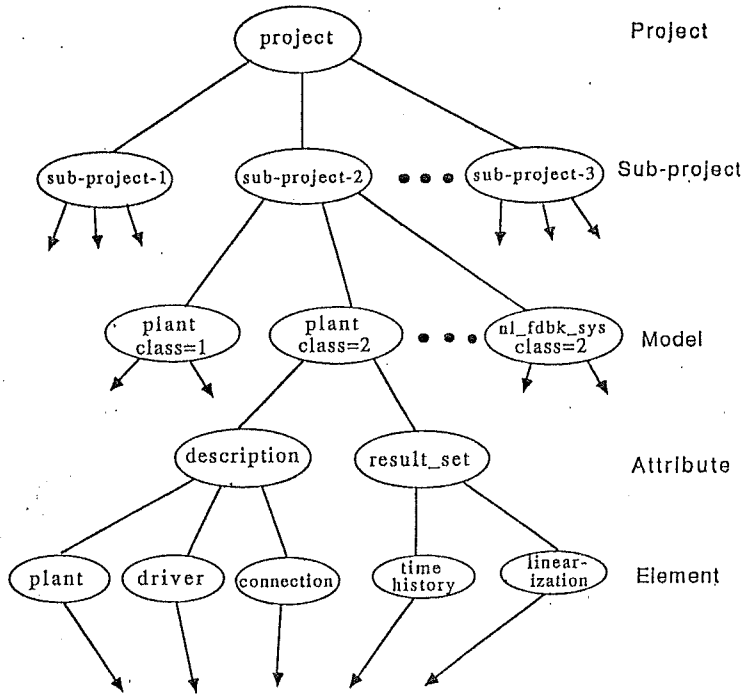

Figure 1. Hierarchical Data-Base Organization for CACE

Description_Relation:	NL_Fdbk_Syst			
Equation_Type:	Nonlinear			
Time_Type:	Mixed Cont / Discr			
Component_set:	Airframe, Comp, Sensor			
Connection_Def:	NL_Fdbk_Conn			
Component: Class:	Airframe Version	Comp Version	Sensor Version	Note
001	001	001	001	Yes
002	001	002	001	No
003	002	003	001	No
004	002	004	002	Yes
005	003	007	002	No

Figure 2. Model Description in a CACE Data-Base

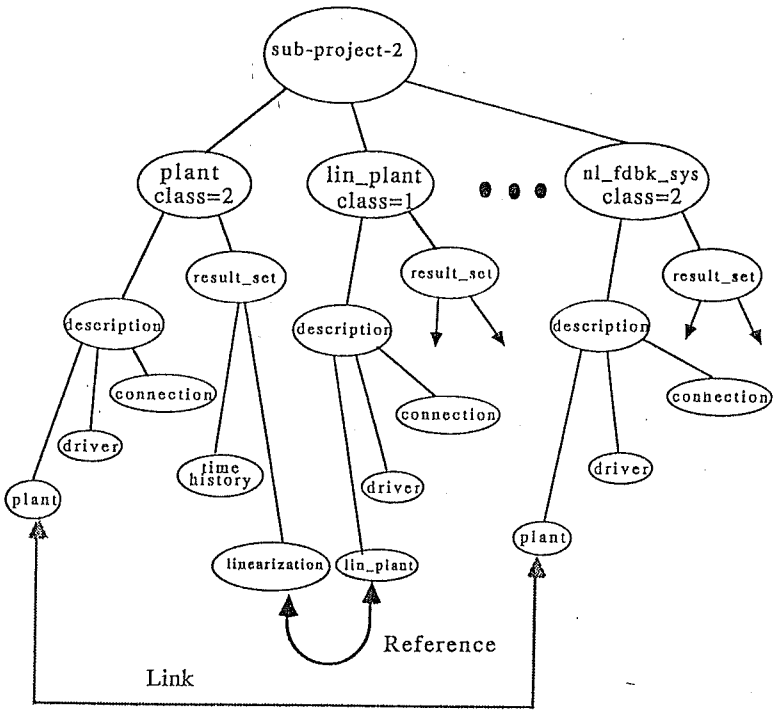

Figure 3. CACE Data-Base with Reference and Link

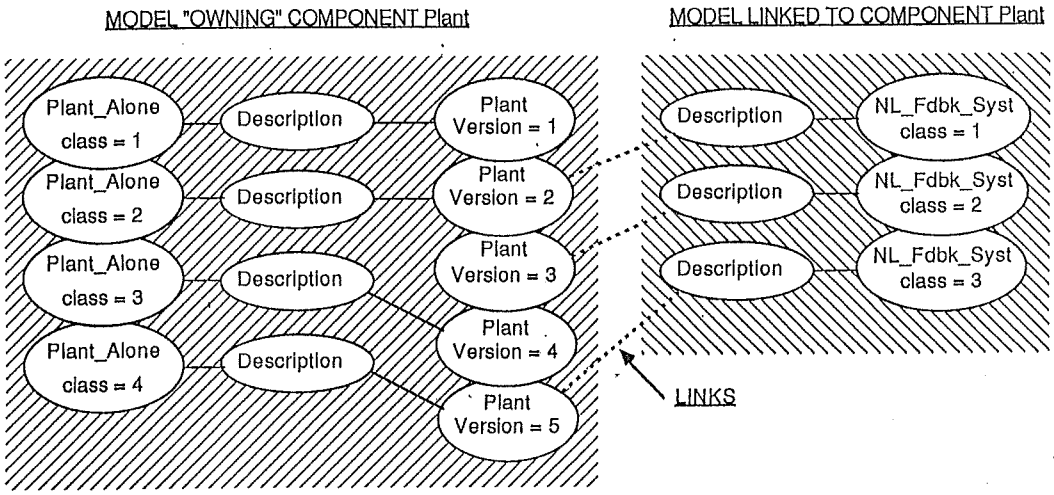

Figure 4. CACE Models with a Linked Component

Result_set_Relation:		model_id = NL_Fdbk_Syst + Class03		
Name	Type	Condition_Spec	Create_Date	Note
Cruise_20k_Mp6	Simulation	CS001	27-Mar-1988 09:44	Yes
Cruise_20k_Mp7	Simulation	CS002	27-Mar-1988 11:17	No
Cruise_20k_Mp7	Trim	CS002	27-Mar-1988 11:52	No
Cruise_20k_Mp7	Linearization	CS002	27-Mar-1988 11:57	Yes
Cruise_20k_Mp8	Trim	CS003	27-Mar-1988 13:06	No

Figure 5. Result_set in a CACE Data-Base