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

AN INTERFACE FOR COMPUTER-AIDED CONTROL ENGINEERING BASED ON AN ENGINEERING DATA-BASE MANAGER

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

Phil McKeehen
Flight Dynamics Labs
Air Force Wright Aeronautical Labs
Wright-Patterson AFB, OH 45433

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

Abstract

A companion paper [1] describes an engineering data-base manager (EDBM) for computer-aided control engineering (CACE). The need for an EDBM to support the complete control system design cycle is also discussed. The obvious benefit of integrating an EDBM into a CACE environment, as demonstrated in [1,2], is knowing how all system models and analysis and design result data are interrelated. This results in a data base that is documentable and reproducible. If the user interface of the system is designed correctly, there can be several major secondary pay-offs as well: information hiding (eliminating the need to recall commands and file names, for example) and direct integration of CACE activity and EDBM functionality. Conversely, if the CACE environment and EDBM are not integrated well, many users will not be motivated to use the EDBM to track models and results.

The features of an EDBM-based user interface to a CACE software environment are described in detail below. This includes demonstrating the execution of all data base operations, including browse, display, edit, purge, delete, and replicate, as well as executing certain CACE activity directly from the EDBM displays. Some of these features have been implemented in a rapid prototype software environment [3,4]; more of them are currently being incorporated in version 1.0 of our CACE environment [5]; a few will remain to be realized in future stages of this work. The primary goal of this user interface design is to make EDBM an integral part of the environment, not merely an appendage.

1. INTRODUCTION

Considerable recent effort in the development of software environments for computer-aided control engineering (CACE) has been focussed on the introduction of ancillary design aids such as expert systems (see [1] for references) and, more recently, engineering data-base managers (EDBMs) [1 - 4]. These efforts have been motivated by the realization that the CACE analysis and design cycle is complicated and difficult to manage without secondary software support [6]. As CACE software becomes broader in scope and more powerful, the need for keeping track of the models, analysis results, control system designs, and validation study results becomes more pressing, especially in situations where the models evolve with time (e.g., are refined as new experimental results are obtained). In a real industrial controls project, the large number of files generated in a complete control system design cycle and the relations among these files can be very difficult to comprehend and manage without EDBM. The most critical aspect of this issue is *data-base integrity,* i.e., knowing which model instance was used to generate each result. This problem can be solved via the coordinated use of a suitable EDBM [1, 2].

Several secondary, less serious, problems can also be addressed via the introduction of an EDBM. In systems without EDBM, the user usually must recall what models are available and know where they are located in storage (e.g., disk, sub-directory, and file name). Results also need to be tracked, both in terms of physical location as well as the conditions under which they were generated. This information is vital in the absence of an EDBM, so that these data elements can be used or displayed by explicit command. A CACE system with EDBM can alleviate this situation as well.

One scheme for an EDBM for CACE has been described in [1, 2]. Very briefly, this system defines a data-base hierarchy having the levels *Project*, *Sub-project*, *Model*, *Attribute*, and *Element*, as shown in Fig. 1. Models are catalogued in a Model_Table. They typically change with time as the control engineer iterates to validate them against empirical results / observations that accumulate over the life of the project and as modeling deficiencies are uncovered and rectified, and therefore have several or many instances. Each model has two attributes: a *Description* and a *Result_set.* Elements of a Description are the primary characteristics, e.g., linear or nonlinear, continuous- or discrete-time, etc., plus the sub-system or component models, e.g., a plant, compensator, sensor, etc. Elements of a Result_set include any data generated with a particular model instance, such as equilibria, linearizations, and simulation results (time-histories) for the nonlinear case or eigenvalues, frequency responses, etc. for linear systems. There are three data elements

corresponding to each entry in the Result_set: the *Condition_spec,* which contains all information required to define how the result was obtained (off-nominal parameters set by the user, choice of algorithm if available, etc.), the *result* itself, and *notes* entered by the user. Elements at the bottom of this hierarchy (model components, results) are all tracked by the EDBM via reference to their storage location (e.g., disk, sub-directory, and file_name); this information is hidden from the user unless it is requested. Each Model in this hierarchy is maintained using a standard version control scheme, so that system models can evolve and the results obtained with each instance of the model can be linked unambiguously with that instance, thus maintaining the integrity of the data base [1, 2].

Rigorous knowledge about how all model and result data elements are interrelated is clearly essential to the control system design cycle. Having this information maintained with integrity guarantees that any analysis and design result can be documented, and results can be reproduced or extended if necessary. To be truly useful, however, considerable care must be taken to design the CACE environment user interface to be compatible with the EDBM, and thus make both the tracking of models and results as well as access to models and CACE operations as simple as possible. This problem - the integration of EDBM into a CACE environment to create a more effective user interface - is the thrust of the presentation that follows.

Outline: Section 2 presents the design approach and principles used to design the user interface to our integrated CACE system; Section 3 illustrates the use of our interface and demonstrates how it integrates EDBM and CACE functionality, and Section 4 provides a summary, comments, and plans for future work in this area.

2. *EDBM-BASED USER INTERFACE DESIGN*

The EDBM organization and functionality [1, 2] has been based on a conceptual model of the control system design process that sees projects and models as being the primary focus for CACE activity. Beyond the basic hierarchical organization outlined above, the data base is quite open and amorphous: There is no closure with respect to the number and types of data elements that may be found in a CACE data base, and the elements themselves are difficult to define and highly variable from type to type. At the simplest level, even linear time-invariant systems may be modeled in a variety of ways (ratio of polynomials *vs* state-space "A, B, C, D"; as separate arrays *vs* packed form [A, B ; C , D], etc.); nonlinear system model representations are completely arbitrary depending on the basic formulation (e.g., $\dot{x} = f(x,u)$) and the simulation language employed by the CACE environment. In the area of results, data elements vary from lists of eigenvalues to arbitrarily large arrays of numbers representing time histories. These features of our CACE data base design (a high-level organization of arbitrary data elements) formed the basis for the decision to access the data base via a "browsing" approach [3, 4] rather than a query language.

The decision to implement a browsing facility to access the data base caused us to see the power of viewing data elements as "objects" to be manipulated in an object-oriented framework instead of a command-driven style of interface. An element is *designated* by moving a pointer to its location in a listing and hitting a 'pick' button or key, then an *operation* is selected by a second move of the pointer and 'pick' operation. Thus, for example, a model in a Model_Table can be designated and then listed, edited, annotated, purged, deleted, or configured for study (loaded into a simulation or analysis package); any data element in a Result_set_Table (time history, frequency response, eigenvalue set, etc.) listed by the EDBM system may be selected and displayed, hardcopied, annotated, or deleted. These examples show that, in effect, the EDBM relation display becomes an object-oriented representation of its data elements, allowing the user to make direct connection with each element on the basis of the appropriate context. Such an approach exploits the obvious interrelations between the EDBM functionalities and standard CACE activities, as the above examples demonstrate, and thus simultaneously achieves direct, immediate access to data-base operations and CACE functionality. In addition, this approach has the corollary benefit of allowing the EDBM to "hide" unnecessary detail from the user - for example, the user does not need to know file names or location, and does not have to learn a set of commands to execute operations.

These ideas served as the basis for defining many elements of our CACE environment user interface. These concepts and the benefits of such an approach are illustrated in the extended examples of this style of user interface provided in Section 3. The features of this EDBM-based object-oriented user interface for CACE are exemplified by showing the execution of all operations on the data base, including display, edit, delete, purge, and replicate, as well as access to CACE operations using such an interface. Many of these illustrations are based on features incorporated in a rapid prototype version of our CACE environment [3, 4].

3. *ILLUSTRATIVE EXAMPLES*

All access to the CACE data base is implemented in the CACE Browse Facility. The two top-level options in our system are Browse_Models and Browse_Results. The sections that follow illustrate the user interface

characteristics pertaining to these functions.

3.1 EDBM Access to Models (Browse_Models)

Entry to the Browse_Models Facility is made via a listing of the project Model_Table. An example of the user's display at the top level is provided in Screen 1.

The first row in this display summarizes the primary nonlinear model for this project, showing that there have been two instances (class = 2, 3) created since the model was first entered, that there is (by definition) no reference associated with this model (a "reference" only pertains to a linear model that was obtained by linearizing a nonlinear model [1], as F-14-Lin001), and that the user has entered notes documenting the two model refinements. The next three rows describe linearized models; presumably they were obtained from the first model, but one would have to check the Reference to determine this for certain and to see which nonlinear system instance was used and which element in the model's Result_set corresponds to the linear model. F-14-Ctrl001 is apparently (judging from the type and user-supplied name) a linear control system; one would have to check the Description to see the details (e.g., which linear F-14 model was used). The last row corresponds to a nonlinear flight control system; again, one would have to check the Description and/or Notes to get all available information (e.g., which instance of the F-14 nonlinear model is used in the control system model).

At this level, several actions available via the 'Action Buttons' portrayed in Screen 1: Display_Description, Display_Reference, Display_Note; Add/Edit_Note, Configure_Model (meaning load the model into a simulation or linear analysis and design package for use), Purge_Model, Delete_Model, and Quit. The user first designates a model by a 'move' and 'pick' key-stroke, then selects the action with a second 'move' and 'pick'. The designation could be based on moving the pointer to the name, in which case the highest class (most recent instance) of the model is designated, or by moving to a specific class number. Most of the corresponding actions listed above are self-explanatory, and thus are not illustrated. Observe, however, that the Configure_Model option represents a major entry point into CACE activity.

One can browse down into a model's data base by designating it and selecting Display_Description; for the last model listed in Screen 1, this results in a component-level display as portrayed in Screen 2. We note that the most important general characteristics are specified at the top, e.g., that the system is nonlinear and mixed continuous- and discrete-time, and that the form is a known special configuration comprised of a plant (the component F-14-Compt, which is presumably also the basic component of the model F-14-Airframe listed in the Model Browsing Screen), linear sensors and actuators, and discrete-time linear (DABCD) compensator. There are the same basic features in the Component_Table as there are in the Model_Table; the main difference is that instances of each component are specified in terms of 'version' rather than 'class'; a given model class corresponds to a specific version of each component which is tracked in the EDBM via the Class_Table provided below the Component_Table. This version control scheme ensures that all results are associated with a given model instance that can be recreated and verified [1]. The linear components listed here do not have references, since they were not generated by linearization. We can edit any component, and thereby create new versions that may be used later to build a new model_class. Note in both browsing tables above that there is an annotation facility (Display_Note, Add/Edit_Note) that allows the user to document the data base as the model evolves, at both the Model and Component level. Again, most of the operations are clear enough without illustration.

3.2 EDBM Access to Result_sets (Browse_Results)

The entry to the Browse_Results Facility is also a display based on the project Model_Table. The user's display at the top level, Screen 3, is similar to the Browse_Models initial screen. This display overviews the entire project, and only indicates which instances of each model (e.g., instances 1 and 3 of model F-14-Airframe) have been used to generate results. The user can now designate a particular model (name + class, by 'picking' on the appropriate class number) and determine specifically what results are available; the second-level Browse_Results display is illustrated in Screen 4.

The elements catalogued in Screen 4 include all results generated using the designated model. These include nonlinear simulations (time-histories), linearizations, trim conditions, etc. Associated with each result one finds a condition_spec identifier (CS_001 etc.) and a note indicator (if there is a 'Y' in the fourth column, then a note has been entered by the user to annotate the data base). As before, there are a variety of obvious operations that the user can apply to each object in the list of results: Display_Result, which will cause the data element to be listed or plotted on the screen, as appropriate; Hardcopy_Result, which captures a result already displayed on the screen; Display_Condition_Spec, which details every off-nominal condition used to generate the result (e.g., setting of a state initial value which will determine the trim condition and corresponding linearized model, perturbation method and size used for linearization, numerical integration algorithm, etc.), the annotation facility commands, and Delete_Result. The 'Quit' button, also shown in previous screens, allows the user to move back up the data base hierarchy, in this case returning to

the Model_Table. In addition, there are several options that provide access to the CACE procedures in the environment: the Edit_Condition_Spec button allows one to modify an existing condition_Spec (e.g., CS_002 → CS_003 by changing parameters or adding commands), and the Generate_Result button can be used to execute a procedure to generate a new result of a specified type with a specified Condition_Spec.

Returning to Screen 3, we find the highest-level access to CACE functionality that is available in the system; this is accessed via the Replicate_Results button. This capability must be used with great care, and only in the appropriate circumstances. Suppose the user creates a new instance of a model (e.g., by editing F-14-Airframe to refine the component model); it may then be desirable to regenerate some or all of the results that were produced for the previous instance. In such cases, the user may execute this capability by hitting the Replicate_Results button. This will bring up the earlier list of results as a menu in which the user can designate the results to be created with the new model instance. Note that this can only be done automatically in cases where the model changes do not invalidate the Condition_Specs used in previous studies.

The illustrations of the Browsing Facility operations are completed by depicting the bottom-level data elements in the Result_set. In Fig. 2 (a) we see a sample Condition_spec, in (b) a Note, and in (c) a Result; these elements correspond to item 5 of the Result_set depicted in Screen 4. Note that this data organization leads to a distinction between CACE operations that *influence* the generation of results and those that *produce* results. An example of the former is specifying the initial value of a state variable before running a simulation; these operations are tracked in the Condition_Spec. Once the user has set up the conditions appropriately, one or more result can be generated.

4. SUMMARY AND CONCLUSION

The objectives of the EDBM-based CACE environment design are:

- to permit the user to organize the data base in a natural framework that reflects the standard CACE design cycle,
- to maintain the integrity of the CACE data base in the face of model evolution via version control,
- to reduce the need to memorize or manually track unnecessary detail via "information hiding", and
- to integrate data-base operations with the rest of CACE activity to the greatest extent possible.

The examples in Section 3 show how these objectives can be met.

Many features of this design were first implemented and refined in a rapid prototype CACE environment [3, 4]. The design presented here is similar but better integrated (accesses greater functionality through fewer screens). A comprehensive version of the CACE system outlined in Section 2 and illustrated in Section 3 is nearing completion [5]. It will include all features described in this paper except Result_set Replication and Condition_Spec Editing; these features will have to be deferred until a later implementation.

Acknowledgement: Development of the EDBM scheme and user-interface concept 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

This work is a part of the MEAD (Multi-disciplinary Expert-aided Analysis and Design) Project [5].

REFERENCES

[1] Taylor, J. H., Nieh, K-H, and Mroz, P. A., "A Data-Base Management Scheme for Computer-Aided Control Engineering", Submitted to *Proc. American Control Conference,* Atlanta, GA, June 1988.

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

[3] Mroz, P. A., "Rapid Prototype Software for Computer-Aided Control Engineering", Master of Science Thesis, submitted to the Graduate Faculty of Rensselaer Polytechnic Institute, November, 1987.

[4] 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 (Contact: K-H. Nieh).

[5] Taylor, J. H. et al - a definitive paper on the MEAD Project, in preparation, 1988.

[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.

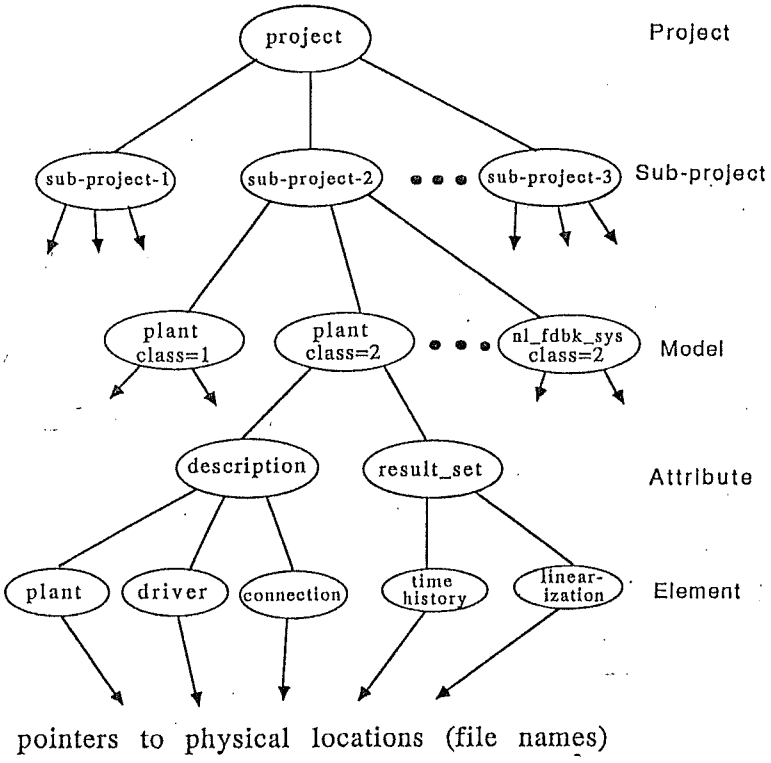


Figure 1. Data-Base Hierarchy for CACE

MODEL BROWSING SCREEN

Models existing in project F-14_PROJ are:

Name	Type	Classes	Create_Date	Last_Mod	Ref	Notes
F-14-Airframe	Nonlinear	1, 2, 3	16-Aug-1987	29-Oct-1987	-	N, Y, Y
F-14-Lin001	ABCD	1	19-Aug-1987	-	Y	Y
F-14-Lin002	ABCD	1	19-Aug-1987	-	Y	Y
F-14-Lin003	ABCD	1	19-Aug-1987	-	Y	Y
F-14-Ctrl001	ABCD	1	23-Aug-1987	-	N	Y
F-14-NL-Ctrl	Nonlinear	1, 2	29-Aug-1987	30-Oct-1987	-	N, Y

Action Buttons:

Display Description | Display Reference | Display Note | Add/Edit Note | Configure Model | Purge Model | Delete Model | Quit

Screen 1. Model_Table Display for Model Browsing

MODEL DESCRIPTION BROWSING SCREEN

Model_Type: nonlinear
Time_Type: mixed_cont/discrete
Model_Form: ctrl_system_with_precompensator

Components in model F-14-NL-Ctrl in project F-14_PROJ are:

Name	Type	Versions	Create_Date	Ref	Notes
F-14-Compt	Nonlinear	1, 2, 3	16-Aug-1987	-	N, Y, Y
Sensor-Set	ABCD	1	22-Aug-1987	N	Y
Actuator-Set	ABCD	1, 2	22-Aug-1987	N	Y
Compensator	DABCD	1, 2	29-Aug-1987	N	Y

Class_Table:

class	F-14-Compt	Sensor_Set	Actuator_Set	Compensator
1	1	1	1	1
2	2	1	1	2
3	3	1	2	2

Action Buttons:

Display Component | Edit Component | Display Reference | Display Note | Add/Edit Note | Quit

Screen 2. Model Description Display

RESULTS BROWSING SCREEN (INDEX)

Results exist for the following Models in project F-14_PROJ:

Name	Type	Classes	Results
F-14-Airframe	Nonlinear	1, 2, 3	Y, N, Y
F-14-Lin001	ABCD	1	Y
F-14-Lin002	ABCD	1	Y
F-14-Lin003	ABCD	1	Y
. . .			
F-14-Ctrl001	ABCD	1	N
. . .			
F-14-NL-Ctrl	Nonlinear	1, 2	Y

Action Buttons:

Display Results | Replicate Results | Quit

Screen 3. Model_Table Display for Result Browsing

RESULTS BROWSING SCREEN

Results for Model F-14-Airframe Class = 3 in project F-14_PROJ:

Name	Type	Condition_Spec	Create_Date	Notes
Cruise_20k_Mp6	simulation	CS_001	16-Nov-1987	Y
Cruise_20k_Mp7	simulation	CS_002	19-Nov-1987	Y
Trim_20k_Mp6	trim	CS_001	19-Nov-1987	Y
Trim_20k_Mp7	trim	CS_002	19-Nov-1987	Y
. . .				
ABCD_20k_Mp6	linearization	CS_001	23-Nov-1987	Y
ABCD_20k_Mp6	linearization	CS_002	29-Nov-1987	-
. . .				

Action Buttons:

Display Result | HardCopy Result | Display Cond_Spec | Edit Cond_Spec | Display Note | Add/Edit Note | Generate Result | Delete Result | Quit

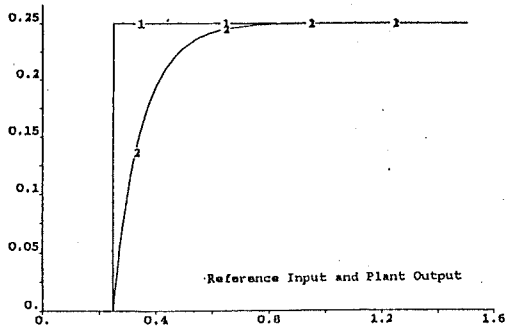
Screen 4. Result_set_Table Display

CS_002

Model = F-14-Airframe, Class = 3

Item	Nominal	Actual
x1init[F-14-Compt]	0.00	1.0
K1[F-14-Compt]	1.00	1.667
Algor	Adams_4	Runge-Kutta_4
Uref[input_gen]	1.00	0.25

(a) Condition_Spec



(c) Result

Wind-tunnel data from 12-14 Sept 1987 processed to produce updated aero-coefficient tables used here

(b) Note

Figure 2. Elements of Result_set Relation Entry 5