Reliability and Quality Control of Production Engineering Computer Programs

S. D. Hansen* and Q. D. McHargt
Boeing Commercial Airplane Company, Seattle, Wash.

Action taken by an aircraft engineering organization to improve the quality and reliability of its technical software is described. Before and after comparisons are given to illustrate that the quality and reliability were improved to where engineers no longer considered them to be a major factor in task planning and the expenditure of programing time for "fire drills" during production use was reduced to a negligible amount. The action taken was administrative rather than technical. It was, however based upon general procedures for control and communication that can be adapted by other engineering organizations for the same purpose.

Introduction

EXTENSIVE use was made of the computer by the Boeing Commercial Airplane Co. during the 1965–1970 period to analyze the 727 and 747 airframes. The use of the computer and finite element analysis methods to analyze the 747 wing-body intersection was the first large-scale commitment made by the Boeing Co. to use these methods at the exclusion of more traditional methods. This commitment plunged computer hardware, computer programs, and the personnel associated with them into a full production atmosphere for the first time. Weaknesses in training, discipline, control, and reliability were exposed which, while noncritical in a development and research atmosphere, proved to be unacceptable in the more rigorous production atmosphere.

Figure 1 is a bar graph of a set of error statistics gathered for one analysis. This particular analysis was a complex substructured finite element analysis that required 687 runs and 83 CDC 6600 central processing hours to complete. The significant conclusions are:

a) Only fourteen percent of the total number of runs would have been required had there been no errors or failures.

b) Thirty-nine percent of the runs, while successfully completed, were later invalidated because of data errors, tape failures, or program bugs.

Some effects of these errors upon production costs and schedules are shown in Fig. 2. The programing and computing costs (not shown) to repair the errors were insignificant when compared to the engineering manpower and schedule costs resulting from the errors. The costs shown include only the effort that had been expended in obtaining the result found to be in error. They do not include time and manpower expended locating the error, determining the effects of the error, and recalling engineering results released containing the error. These and other similar experiences resulted in recognition by engineering management that:

a) The low reliability of existing technical software being utilized in production engineering applications was a major contributor to schedule overruns and high computing costs.

b) The quality of new technical software being released for engineering applications was well below production requirements.

A study was made to determine a positive course of action to establish adequate methods to insure computing reliability.

Study Results

The study showed inadequate reliability of technical software originated from: a) lack of training of the engineering staffs and their management in the application of computer-based technology; b) lack of release and change controls; c) lack of an effective communication media between those developing and maintaining the computer programs and those using them; d) lack of systematic checkout and validation methods; and e) lack of effective documentation definitions and requirements.

The action taken to correct these deficiencies is outlined in the following paragraphs.

Training

Training programs were developed for engineers and supervision. The intent was to develop a general philosophy of computer applications and a level of skill within the engineering staffs that would, in their daily work: a) reduce the volume of input data errors and output interpretation errors to acceptable levels, and b) reduce the volume of misapplications of the computer and provide some guidance for including the computer in early product development planning.

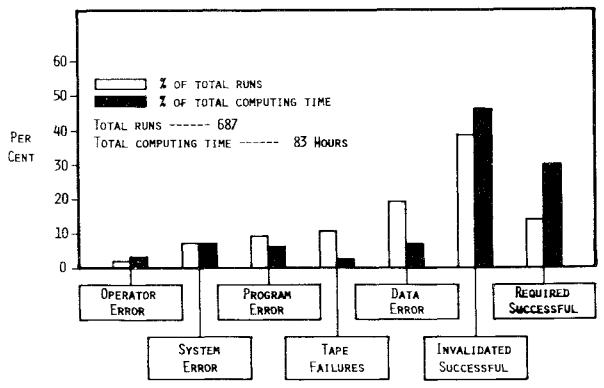

Fig. 1 Wing-body finite element analysis—processing statistics.

Presented as Paper 73-356 at the AIAA/ASME/SAE 14th Structures, Structural Dynamics, and Materials Conference, Williamsburg, Va., March 20–22, 1973; submitted May 18, 1973; revision received February 11, 1974. Projects of this kind require the active participation of many people. The authors acknowledge the work of P. Southgate and N. Prewitt in developing the methods; of R. Palmer, Head of Technology Systems Unit, representing computing management; and of R. E. Miller, Head of Stress Analysis Research for effecting the implementation.

Index categories: Computer Technology and Computer Simulation Techniques; Reliability, Quality Control, and Maintainability; Aircraft Structural Design (Including Loads).

*Currently Systems Requirements Manager, SAMA Division, Boeing Computer Services.

†Computer Program Configuration Control, Structures Staff.

Error in Computer Code	Engineering Man-Months Lost	Months Flow Time Lost
Beam shear areas reversed from that given in data	2	½
A factor of ½ missing in applying substructure interaction forces to centerline nodes	8	2
Beam skin line node to centroid offsets reversed from that given in data	15	7

Fig. 2 Examples of time lost due to errors in computer code.

Supervisory Training Course

The supervisory training course emphasized skill building in the following subjects:

a) Computer Technology

1) Matrix algebra formulations of simultaneous equations, eigenproblems, and structural elasticity problems.
2) Computer hardware operations including central memory, central processor, timing, control peripherals, and job processing.
3) Programing lauguages and their use to develop the basic hardware capabilities into useful computational and data handling tools.
4) Basic design of large integrated analysis computer systems, including matrix formulated structural mechanics, development of finite elements, high level program control languages, and basic construction and coupling of the system modules.

b) Task Planning

1) Analysis objective and review criteria.
2) Computing resources and technical skills required, including an evaluation of existing computer programs to determine adequacy, reliability, or needed modifications.
3) Task flowcharting and scheduling.
4) Basic methods of idealizing the structure for a computer-based analysis.

c) Review and Control Requirements and Procedures

1) Computer program reliability.
2) Preparation and control of data.
3) Quality control of the computer output including automated checks on equilibrium and compatibility, review of the results and idealization in relation to the actual structure, and use of plots and print formating to aid rapid accurate interpretation.

d) Interactive Computer Accessing

1) The time-share concept.
2) Flowtime reductions.

Engineering Training Course

The engineering training course was directed explicitly to developing daily skill in the use of the computer in analyzing aircraft structures. The course was divided into sections depending upon the skill of the engineer. Engineers and supervisors were encouraged to complete the series.

a) Basics: Matrix algebra, systems of linear equations, eigenproblems, computers, and FORTRAN language.

b) Mechanics: Matrix formulated structural mechanics and finite element analysis.

c) Structural Applications: In-class applications of existing computer programs to actual structural analysis problems.

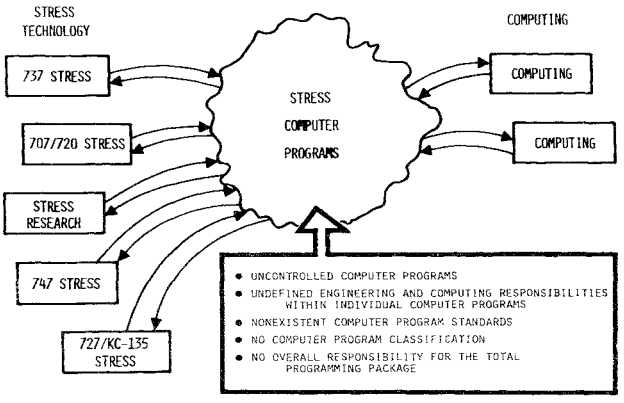

Fig. 3 Before release and change control.

Release and Change Control

There are several hundred structural engineers in the various Boeing Commercial Airplane Co. facilities in the Seattle area. Many of these engineers use the same computer programs. Several problems existed at the time release and change control procedures were instituted.

a) There was no distinction between development and production versions of programs.

b) Program versions were released directly to engineering by the individual programer currently working on the program.

c) There was no visible distinction between versions of programs in naming or serial numbering.

d) There was no procedure to maintain the integrity of a production version, i.e., to prevent tampering with the source code.

e) There was no control to ensure that changes being made to the programs were in the best interest of the users and not just the inventive interest of the programer.

f) There was no facility to preserve versions of programs over long periods of time or to purge obsolete versions.

g) No individual, in programing or engineering, was specifically responsible for the integrity of production computer programs.

This situation is generally illustrated in Fig. 3.

To overcome these deficiencies, a release and change control procedure was established with the following characteristics:

1) A single individual was assigned program control responsibility. His function was to ensure availability and provide change control of released production programs.

b) All releases of production computer programs are made through him.

c) He assigns program names and version numbers and creates master and production tapes.

d) Program names are of the form S1234A3, where "S" signifies the technology such as "stress," 1234 is a serial number, A is a version name associated with a capability change, and 3 is a version number associated with a correction to A but not a change in capability.

e) Master tapes contain the following files:
File 1—all source code statements
File 2—relocatable binary compilation
File 3—absolute binary compilation
File 4—various compiler outputs
File 5—input test data to prove compilation only
File 6—output from the test data
File 7—catalog.

f) Production tapes contain two copies of both the relocatable and absolute binary files. Source code is not released on a production tape.

g) Master tape numbers are not published. They are, however, available for reading by either engineers or pro-

RECORD NO.	FIELD LENGTH	PACKAGE	CHKSUM	TIME	DATE
1	48407	S0172BF	711	00:01	26/02/70
2	7190	GENRAT	413	00:01	26/02/70
3	29907	INFO	2031	00:01	26/02/70
4	20447	PLATE	2623	00:01	26/02/70
5	10563	BEAM	213	00:02	26/02/70
6	132	MERGE	4760	00:02	26/02/70
7	[illegible]	CONMER	[illegible]	00:02	26/02/70
[illegible]					
16	40297	CINCIG	4324	00:02	26/02/70
17	40417	PUTOUT	7567	00:02	26/02/70
18	59228	STRESS	4573	00:02	26/02/70
19	55346	REPART	3612	00:03	26/02/70
			TOTAL CHECKSUM IS		5611

Fig. 4 Example of catalog.

gramers who have legitimate reasons. Production tape numbers are published and the same number is maintained indefinitely. If a production tape fails, a new tape is generated and the old number transferred to the new tape. This eliminates any need to continuously inform users of changing tape numbers.

h) Once released, a production tape is never changed. Error corrections are made by incrementing the version number and releasing a new set of tapes. Old versions are maintained until, 1) a trace of usage statistics indicates usage has fallen to zero, 2) a later version contains the capability of the older version, and 3) a published intent to purge results in no objection from the using community.

i) Changes to controlled programs are made by the following procedure: the programer or engineer making the change works from a copy of the source code from the master tape. When the change is ready for release, only the change cards are given to the person assigned control responsibility. This person processes the update cards and source code statements from his master tape to obtain a new master tape and production tapes. A catalog of both the old and new versions is created and cross checked to ensure that only the intended changes have been made.

j) A catalog of the source, relocatable, and absolute files is made. An illustration of this catalog is given in Fig. 4. Basically, the catalog forms a checksum of each record on the file and gives the date of compilation. It also gives memory field lengths and an over-all checksum. Whenever the legitimacy of a program version is in question, a catalog is taken and compared with the master tape.

The release and change control procedure is generally illustrated in Fig. 5.

Communication Media

A publication was created to 1) provide inventory of computer programs, and 2) furnish using engineers with sufficient information to identify and use a particular computer program.

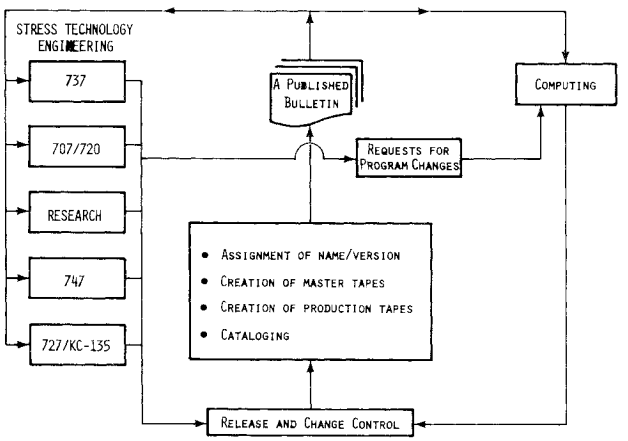

Fig. 5 With release and change control.

This publication contains: a) the name and version of the program as described earlier; b) a descriptive or mnemonic name; c) the name and telephone number of the engineering and programing supervisors assigned responsibility for the program; d) references to all documentation released regarding the program; e) tape numbers, tape configuration, field lengths, and all other information necessary to run the program; and f) a short abstract describing the area of applicability and distinguishing between this version and other versions.

Announcement of intent to cancel or purge a program from the inventory is made in this bulletin along with other general information regarding program development, control or use. The bulletin was originally published on paper. It has recently been placed on a permanent file on the computer and may be accessed from a terminal.

Systematic Validation and Checkout

Checkout methods have typically taken one of two extremes. Either they consist of running a few typical cases through the program or they consist of such extensive and complicated procedures that they are never implemented. It was not uncommon to find large checkout decks being used without any knowledge of the relationship between the test decks and the program logic. As a result, the production programs frequently contained errors such as those in Fig. 2. It was not unusual to find that newly released programs would only execute the checkout data. To overcome this deficiency, a rational set of requirements were established.

a) Validation of the engineering theory is separated from validation of the program logic. Test cases establishing the validity of the theory are in nowise accepted as proof of the program logic except for the particular case of those test runs.

b) Logically isolated modules of the program are identified. Hence, test cases considering coupling of these modules can be eliminated.

c) There is a known relationship between the test data and each item being checked.

d) The test runs and the particular logic element being tested are documented for review by users and future modifiers of the program as illustrated in Fig. 6.

Figure 6 is an example of the program logic tests for a program recently developed at Boeing. This particular program has about 7,000 executable source statements. Seventeen pages of checkout cases similar to that shown in Fig. 6 were identified. Although the test cases were developed and documented as the program developed, it is estimated that about two months out of an eighteen month development time was used for developing and executing the test cases. In some instances, the test was made by inspection. This type of check can be very effective where the specific item to be checked has been precisely identified.

ROUTINE	PURPOSE	CHARACTERISTICS	TEST
SBLKDT	SBLKDT COMPUTES THE DETERMINANT OF A SYMMETRIC MATRIX WHICH HAS AN OVERLAPPING AND REPEATED SUBSTRUCTURE.	SYMMETRIC MATRIX WITH 1. START SUBSTRUCTURE ONLY 2. START AND END SUBSTRUCTURES ONLY 3. START,1 REPEAT AND END SUBSTRUCTURES 4. START,2 REPEAT AND END SUBSTRUCTURES	F-1 F-38 F-8 F-22
		1. [illegible]	F-1
	[illegible] STIFFNESS MATRIX	3. START,REPEAT AND END SUBSTRUCTURES	F-22
STRMOV	MOVES CHARACTERS STARTING AT A POSITION IN ONE ARRAY TO A DIFFERENT POSITION IN ANOTHER ARRAY.	1. STARTING ARRAY IS MORE THAN ONE WORD 2. RECEIVING ARRAY IS MORE THAN ONE WORD	INSPECTION F-37

Fig. 6 Example of validation and checkout documentation.

Documentation

Documentation guidelines were established in which contents and intent were emphasized and specific formats deliberately avoided. Four subjects were identified: theory, coding, testing, and use. Whether these four subjects are treated in the same document or as separate documents is entirely a function of size and need. The guidelines included the following.

Theory

This document contains derivations, explanations, illustrations, limitations, recommended applications, and validation of the theory. The purpose of this document is to provide engineers and programers with a precise statement of the theory upon which the program is based. There should be a one-to-one correspondence between this document and the program code.

Coding

This document describes the design of the computer program and provides adequate information to maintain and modify the program with reasonable efficiency after the original developers have relinquished responsibility. The primary mode of documentation is macro flowcharts and comments in the source code. Detailed flowcharts largely duplicate well-commented code and are not used. Detailed guidelines for commenting source code were developed and generally included the following information.

a) After the Program Header

1) program name
2) a log of persons developing and modifying the program complete with a statement in general terms, of the modifications made and date
3) an abstract describing the program purpose and capabilities
4) a dictionary of definitions for all variables generally used throughout the program. Particularly, a correlation between program variables and theory variables is to be made here
5) system library routines used
6) data organization and storage requirements.

b) Subroutines

1) purpose
2) statement of method or a reference to the method
3) list of variables and definitions.

c) General

1) Any comments which lend clarity to the code
2) explanation of any variable used locally where usage is not obvious from the code.

An advantage of this form of documentation is that each listing contains the documentation relevant to it. Another advantage is that, once convinced of the merits, programers can enter comment cards much more efficiently than they can update and modify typewritten documents.

Testing

This document generally contains the validation and checkout described earlier in this paper. Its purpose is to demonstrate the degree to which the logic of the program was validated and to provide an index to test decks for those who maintain and modify the program.

Use

This document should be primarily a reference text for the user. If written correctly, it will relieve both the user and developers of the necessity of frequent communication. Only those program capabilities included and validated in the code should be described.

The document should contain: a) a summary of program objectives and capabilities, b) a summary of the theory, c) a summary of program design, d) access to the operating system, e) program control, f) program data formats and descriptions, g) output formats and descriptions, h) program limitations, i) error diagnosis and recovery, j) estimates for run times and input and output data volumes, k) relationships to other programs, and l) examples.

Implementation

Organizations the size of the Boeing Engineering staffs have considerable inertia and place great credance in traditional proven methods. Several actions were taken to achieve a successful implementation:

a) Program Inventory

An inventory of computer programs was made. The programs were classified into three groups.

1) *Controlled:* Programs in current use containing a capability to be maintained indefinitely.
2) *Obsolete:* Programs in current use but containing capability duplicated in other programs or of questionable long term value.
3) *Uncontrolled:* Programs for which current tapes or decks could not be located and which appeared to be receiving no use.

Tapes and decks were collected for the first two categories and master and production tapes produced. The publication described earlier was released. During the first year, all programs were either moved into the controlled category or purged from the inventory.

b) Documentation

A special task force was assigned to create documentation for the more important programs. Documentation of new programs being developed was upgraded.

c) Modifications

Program modifications were collected into packages and implemented in time periods of three to six months. Emergency error corrections were still made but only as necessary to comply with product schedules.

d) Monitoring

Many private versions of the programs brought into the inventory existed among the engineers and the programers. The following actions were taken to eliminate these versions and shift production running to the controlled versions:

1) The name of each program processed was picked up by the computer operating system and recorded along with budget number, name of person submitting the run, tape number, and other identifying information.
2) A report was formed each month by scanning the data collected by the operating system. Since the legitimate program name was inserted in controlled production tapes and not in unauthorized versions, it was not difficult to identify program versions outside the controlled set.
3) Engineers using the uncontrolled versions were asked to shift their work to a controlled version. Tapes or card decks of uncontrolled versions were released or destroyed. The number of times particular programs were used was also given in the report, forming the basis for purging unused programs.

e) Training

Essentially all affected supervisors were familiarized with the control procedures and purposes through personal contact and through the training courses described earlier. The transition from uncontrolled to controlled was scheduled over a period of one year. The procedures were first established in the Stress Unit and are now being extended to other technologies.

Results

Specific results from this type of effort are difficult to quantify. However, some trends and specific instances have been noted.

a) Coding errors occurred in about one out of every ten executions for the SAMECS program, a large finite element program used for substructure analysis of the 747 and SST airframes, prior to imposition of the controls. A few months after the imposition of the controls, the failure rate had dropped to about 1 coding error in 300 executions.

b) A comprehensive new solution module (about 25,000 source cards) was placed into the SAMECS program in July 1970. This solution module interfaced extensively with existing code and was validated using the method described in this paper. From July through November, only two coding errors occurred despite heavy using of between 250 and 350 executions each month.

c) A computer program developed under contract to NASA Langley was validated using the method described in this paper. The program has in the order of seven thousand executable statements. It executed on their computer installation without modification and has performed in daily to biweekly usage with only ten coding errors over a fifteen month period. Several new versions of the same program and additional programs have been delivered with the same result.

d) The program inventory was reduced by over 50%. This reduction released considerable tape and file storage.

e) Time formerly spent by programmers in counseling users and exercising "fire drills" has been reduced from around 50% of their total working time to around 10%.

f) The use of production engineering work to check out new programs or modifications to existing programs has been essentially eliminated.

g) Engineers are assured that they can return months afterwards to find a program still intact and possessing the same integrity as when last used.

h) One major operating system change has occurred. Programs under control were converted essentially without incident and without interruption to ongoing production work.

Conclusions

The procedures given in this paper have been effective in upgrading the reliability and quality of technical software within the Boeing Commercial Airplane Co. engineering staffs. Although care must still be exercised when committing production work to the computer, the amount of manpower and schedule time lost is only a fraction of the former condition. It would not be correct to imply that the improvements have come totally from these procedures. People and programs have matured and computer hardware has become more reliable. However, these procedures introduced discipline into the administration of program development, checkout, release, and modification. Significantly, the manpower required to control about 50 stress technology computer programs is about one quarter of one man. The procedures are largely self-administering, both engineers and programers having recognized the efficiency and convenience of the process.

Aside from the preceding, a principal benefit has been increased productively resulting from a better understanding of the computer by management and engineers, the removal of irrelevancies in engineering and programing work assignments, and a disciplined but unoffensive method of obtaining and maintaining computer program integrity.

APRIL 1974 J. AIRCRAFT VOL. 11, NO. 4

Incremented Flutter Analysis

R. F. O'Connell*
Lockheed-California Company, Burbank, Calif.

A method is presented to determine the magnitude of any particular increment (mass, stiffness, damping, etc.) necessary to satisfy prescribed flutter constraints. A reduced-order eigenvalue problem is formulated from the basic flutter equations, and the required increment is determined to the degree of accuracy inherent in the basic flutter equations. Application of the procedure to the evaluation of flutter with external stores is presented, as well as a simplified stiffness optimization procedure. Use of the method in an interactive (computer graphics) mode is described.

I. Introduction

DURING the design and analysis of an aircraft structural system, it is frequently desired to evaluate the effect of parameter changes on the flutter characteristics of the system. The modified parameters may be inertia, stiffness, aerodynamics, etc., and may represent a variety of external stores, stiffness configurations, mass ballast arrangements, or other items of interest. If the number of

Presented as Paper 73-392 at the AIAA/ASME/SAE 14th Structures, Structural Dynamics, and Materials Conference, Williamsburg, Va., March 20–22, 1973; submitted April 23, 1973; revision received February 25, 1974.
Index categories: Aircraft Structural Design (Including Loads); Structural Designs, Optimal.
*Aeromechanics Department Engineer. Associate Fellow AIAA.