AN EXAMPLE OF SOFTWARE QUALITY ASSURANCE TECHNIQUES USED IN A SUCCESSFUL LARGE SCALE SOFTWARE DEVELOPMENT

Paul J. Kacik
Hughes Aircraft Company

1. INTRODUCTION

Development of the software package for the Combat Grande Air Defense System was considered by the Hughes Aircraft Company to be highly successful in that a reliable system was produced that met customer requirements and was completed within time and budget allocations – a feat not often attained in large scale software developments. Much of the success can be attributed to the software quality assurance (QA) techniques used. Some of these QA techniques are listed in Table 1 along with the phases in which they were used.

Table 1. QA Techniques

QA Techniques	Software Production Phase	Integration Phase	Formal Test Phase
Software Simulation Model	X	X	
Extensive Design & Code Review	X		
Computer Simulator	X		
Independent Test Team		X	
Simulation Exercise Generator	X	X	X
Data Reduction Program	X	X	X
Timing Performance Monitor		X	X
Error Correction Verification Team	X	X	X
Special Analysis Programs			X
Traceability Matrix			X

This paper describes these QA techniques in some detail, as well as those aspects of the system and software development program that permitted these techniques to be used effectively.

Background information is presented first which describes the system, software, organization and software configuration management. This is followed by a description of the three major phases of software development. The overall results are then presented, followed by recommended improvements and conclusions. Many of the QA techniques listed in Table 1 were used in several phases of software development. However, a particular technique is discussed only in the phase in which it was most extensively used.

2. BACKGROUND

Combat Grande is an air defense system built for Spain and jointly funded by the U.S. and Spanish governments. The contract was awarded in 1974, and the system became operational in 1977.

2.1 SYSTEM DESCRIPTION

The Combat Grande system is comprised of a Combat Operations Center/Sector Operations Center (COC/SOC) and a number of long range radar (LRR) sites. Figure 1 illustrates the system configuration in a manner which highlights the computers. A Hughes model H5118M central computer connects to three satellite minicomputers. The first minicomputer processes information exchanged with remote radar minicomputers. The second minicomputer controls peripheral devices such as a line printer, card reader, etc. The third minicomputer interfaces with a console subsystem which consists of a large group of manned display consoles, each of which is driven by a console controller minicomputer.

2.2 SOFTWARE DESCRIPTION

The software consisted of 680,000 instructions. Of these, 395,000 instructions were developed and verified on previous similar projects, and for the most part consisted of support programs such as the JOVIAL compiler. The remaining 285,000 instructions were newly developed for Combat Grande.

Programs for the central computer were coded in JOVIAL, whereas programs for the minicomputers were coded in assembly language.

2.3 ORGANIZATION

An organizational framework was established which consisted basically of an application programming section, a support programming section and a software test section. Initial staffing was made primarily with highly experienced personnel obtained from other command and control projects whose software development

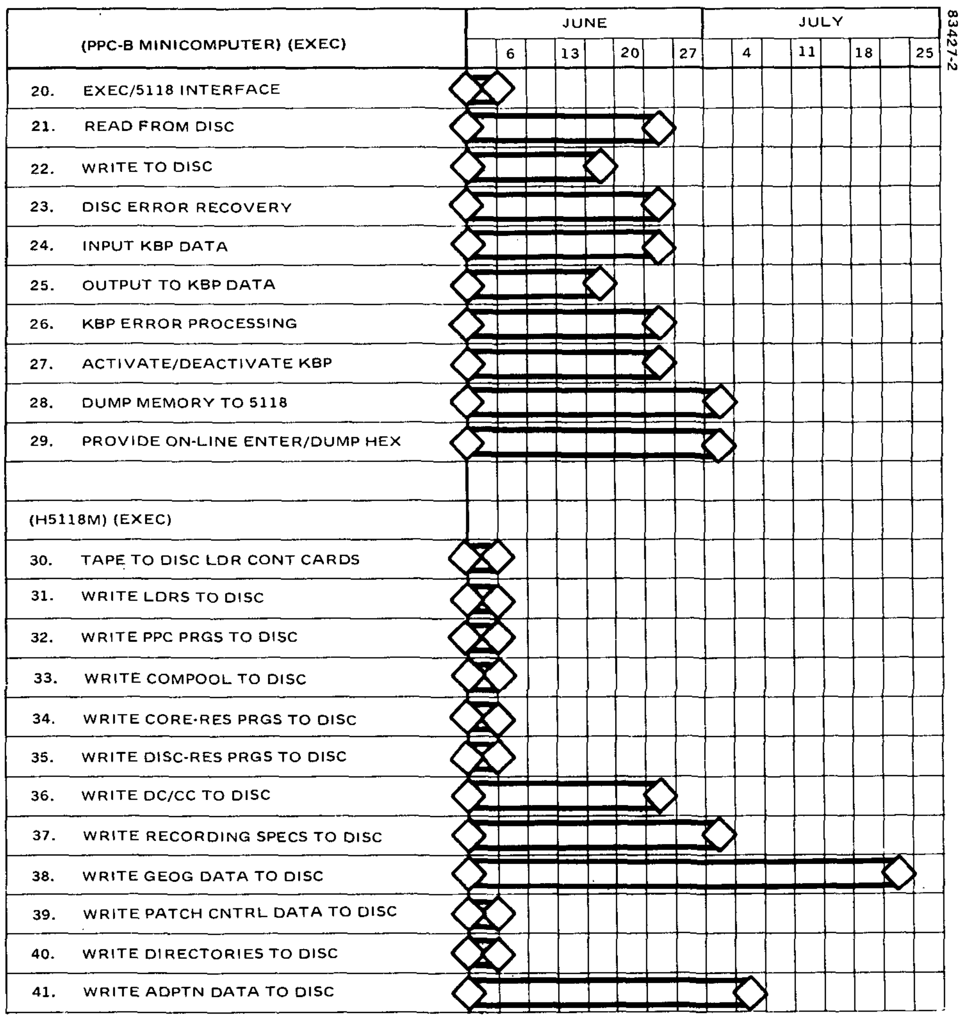

Figure 2. Portion of Integration Schedule

4.2 TAPE VERIFICATION (cont)

When the verification team completed its activity, the integration team verified the new software contained on the master tape using test materials and procedures previously developed by the independent test group. Verification consisted of running a small subset of the test sequences from the procedures.

Both teams would attempt to correct any problems encountered. Those problems which could not be corrected were neutralized in some manner prior to releasing the tape for further testing.

Present thinking generally favors the use of a written regression test during tape verification to thoroughly check previously tested software for new errors. However, counter arguments have been presented which claim that the amount of redundant testing presently accomplished in the various test phases renders regression testing unnecessary. In attempting to evaluate the effectiveness of regression testing as a QA technique, a deficiency in the PTR reporting procedure was discovered. The cost effectiveness of regression and other testing could be reasonably estimated if statistics were available on the error categories and the test phase of error detection. This data was not reported on the PTR forms.

4.3 INDEPENDENT TEST

This phase of testing was perhaps the most effective of the QA techniques used in that voluminous testing was accomplished in an organized and objective manner. It was performed by a test group which was independent from the personnel involved in program development. Test plans and detailed procedures were written to verify each requirement of the performance specification. Tests were run with the equipment and software in as close to an operational configuration as possible. Test procedures were designed to accommodate the incremental availability of the software. By creating many small tests, the impact of software schedule changes to the overall independent test schedule was minimized. Ninety-three tests were conducted. Each test was repeated until all program errors were corrected and verified and the test ran error free.

Several QA tools were used. A simulation exercise generator was used extensively to provide simulated radar message inputs. The messages were generated using an off-line program and derived from a scenario entered by the user. A recording program was used to selectively record data base tables in real time. A flexible data reduction program was used to select and reduce the recorded information into convenient formats and units for analysis. Each of these tools was designed such that a detailed knowledge of the software being tested was not necessary for its effective use.

Customer approval was not a requirement for independent tests. The customer maintained visibility by reviewing the test plans, test procedures, and test reports, and by observing selected test runs.

Avoiding the overhead associated with formal documentation review cycles and formal witnessing protocol permitted the voluminous testing to be accomplished with a minimum number of test personnel.

During independent testing, the timing simulation model was replaced by an on-line performance monitor which measured actual program timing under a predefined processing load.

5. FORMAL TEST PHASE

The formal test phase consisted of in-plant formal qualification testing (FQT) and on-site system testing.

5.1 FORMAL QUALIFICATION TEST

The purpose of FQT was to verify to the customer that the software met all performance requirements prior to shipment of the system to the customer's facilities. Test plans and procedures were written and negotiated with the customer. Thirty-seven formal qualification tests were conducted. Each test was witnessed by customer representatives.

Dry runs for FQT were performed while independent tests were still in progress. In many cases FQT tests duplicated the verification performed by the independent tests (although FQT were less thorough and more statistically oriented). Nevertheless, enough software errors were detected in FQT dry runs that the two types of testing should be considered complementary rather than redundant.

Several special test programs were developed for statistical analyses required in the tests. Use of these programs avoided tedious and repetitive hand calculations.

A traceability matrix, called the Verification Cross Reference Index (VCRI), was used to insure the verification of all specification requirements.

5.2 SYSTEM TEST

System testing was undertaken with the equipment and software installed in its final configuration and environment. Initial testing used simulated inputs to verify system readiness. All subsequent testing used live inputs from the remote radar sensors. Customer operations and maintenance personnel were used in the latter stages to test the system in an operational mode. Over 100 tests were completed with relatively few software errors discovered.

During the early portion of system testing, operator training courses were conducted for customer personnel. During this training, novice operators, using unexpected and unusual approaches, uncovered a completely new set of software errors. This led to the conclusion that using the operational software to support a training course is an efficient QA technique. It is apparent that a QA technique was required which would enhance more thorough testing of the software, resulting in the early detection of this type of errors.

6. RESULTS

As mentioned previously, the software was produced within time and budget allocations. Final system sell-off occurred on schedule.

About 3000 software problems were reported prior to system sell-off. Of these, about 250 were reported during the on-site system test phase. Figure 3 illustrates the reported software problems as a function of time and test phase.

The number of software errors reported during the system test phase seems large at first inspection. However, the number of errors was less than that expected based on the experience of similar previous projects. The following factors contributed to the errors found during this phase:

a. Software interface with the radars and communication equipment was established for the first time.

b. The system was placed in an operational configuration for the first time.

The basic stability of the software was evidenced by its efficient support of the very heavy system test schedule mentioned in the previous section. This stability was a positive indication of the effectiveness of the QA techniques used in the prior development phases.

7. RECOMMENDED IMPROVEMENTS

The basic QA approach used on Combat Grande proved itself sound and has already formed the basis for future large-scale software developments at Hughes-Fullerton. However, the use of more automated tools would have been beneficial. Hughes-Fullerton is in the process of increasing its repertoire of automated QA tools and techniques for use on present and future projects. Based on the experiences of the Combat Grande project and other similar projects, a list has been established of additional required QA tools and techniques judged to be the mose promising for the near term. They are currently available or in the process of being developed or acquired. The following is a partial list:

a. Complexity/Path Analyzer. An automated tool which processes source code to calculate program complexity, determine optimum test paths, and insert probes to support monitoring of thoroughness of testing. This tool will support all phases of testing up to and including independent test. The test goal of exercising all program instructions and all decision options will be enhanced by the use of this tool. More errors of the type described in 5.2 should be detected during early test phases.

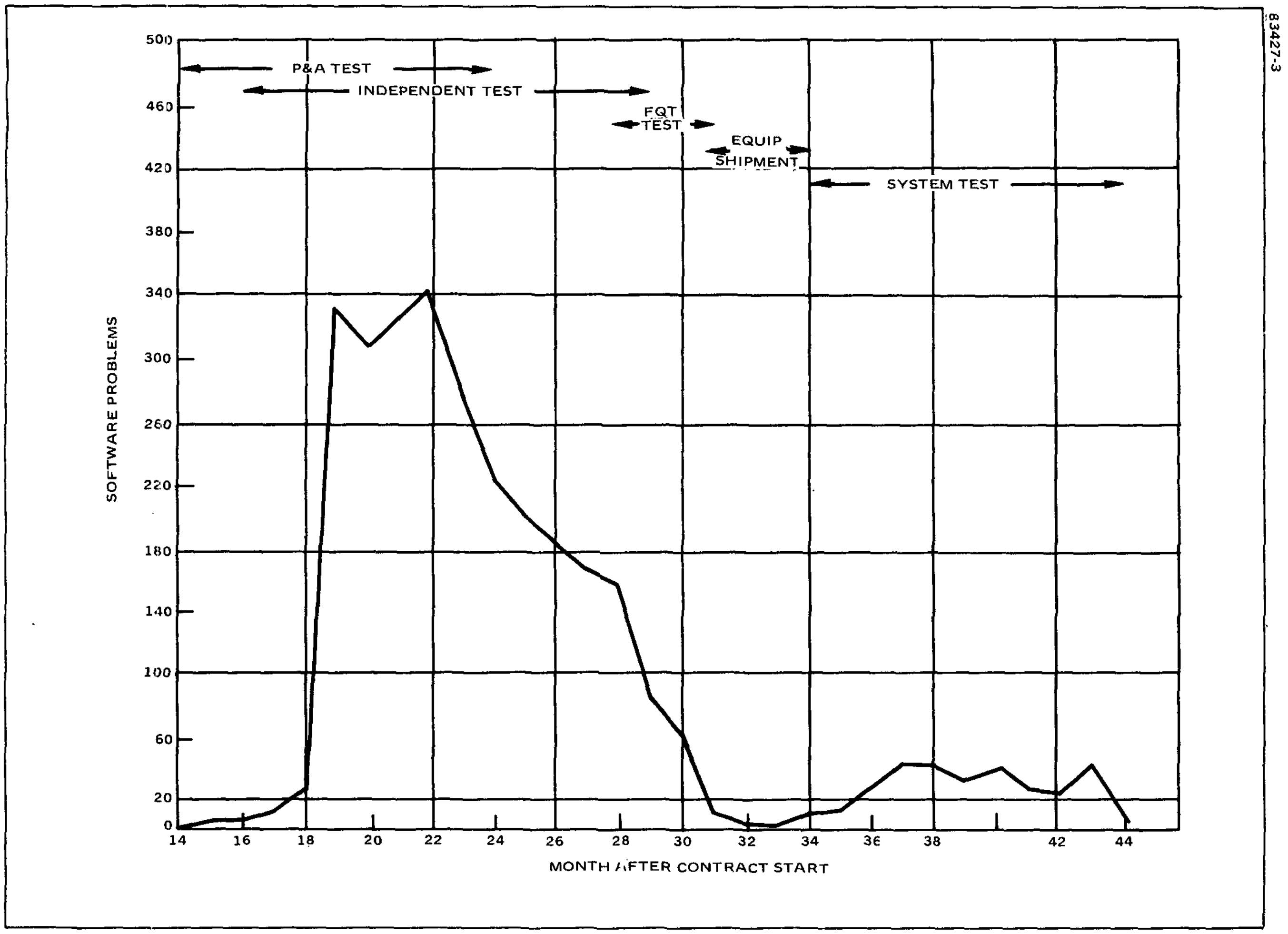

Figure 3. Number of Problems Reported Each Month

b. AN/UYK-40 Simulator. A simulator for the AN/UYK-40 minicomputer is being developed. It will be resident on the IBM-370 computer and will have an extensive debug package. The benefits of the tool have been discussed previously in 3.3.

c. Regression Testing. Regression testing has been used in some projects in the past. It will be reinstituted on present projects and evaluated for cost-effectiveness as discussed in 4.2.

d. Modified PTR Reporting Procedures. Although not strictly a QA technique, the collection of statistics on error categories can be used in evaluating the effectiveness of the various QA techniques, and thereby allow the optimum amount of effort to be applied to each technique. An example is the evaluation of regression testing (c. above). Additional data categories have been added to PTR/PCR forms.

8. CONCLUSIONS

No single factor can be identified as the key to the effective software validation achieved on the Combat Grande project. However, the experience proved that a combination of well known quality assurance techniques can be used effectively to enhance the success of a large scale software development. Continued improvements can be made, especially in the use of automated QA tools.