Defence Science Journal, Vol 47, No 1, January 1997, pp. 27-38
© 1997, DESIDOC

Assuring Quality & Reliability in Complex Avionics Systems Hardware & Software

V. Haridas

Centre for Military Airworthiness & Certification, Bangalore - 560 037.

ABSTRACT

It is conventional wisdom in defence systems that electronic brains are where much of the present and future weapons system capability is developed. Electronic hardware advances, particularly in microprocessors, allow highly complex and sophisticated software to provide high degree of system autonomy and customisation to mission at hand. Since modern military systems are so much dependent on the proper functioning of electronics, the quality and reliability of electronic hardware and software have a profound impact on defensive capability and readiness. At the hardware level, due to the advances in microelectronics, functional capabilities of today's systems have increased. The advances in the hardware field have an impact on software also. Nowadays, it is possible to incorporate more and more system functions through software, rather than going for a pure hardware solution. On the other hand, complexities of the systems are increasing, working energy levels of the systems are decreasing and the areas of reliability and quality assurance are becoming more and more wide. This paper covers major failure modes in microelectronic devices. The various techniques used to improve component and system reliability are described. The recent trends in expanding the scope of traditional quality assurance techniques are also discussed, considering both hardware and software.

NOMENCLATURE

A	Temperature cycling
B	Random vibration
C	High temperature
D	Electrical stresses
E	Thermal shock
F	Sine vibration, fixed frequency
G	Low temperature
H	Sine vibration, sweep frequency
I	Combined environment
J	Mechanical shock
K	Humidity
L	Acceleration
M	Altitude.

1. INTRODUCTION

1.1 Quality & Reliability

There are many definitions of quality. All these definitions are somewhat different, but all these agree that the customer's satisfaction is the goal. If the customer can be satisfied for every aspect of the product (such as performance, user documentation, field support, availability or service, etc), the quality may certainly be rated as excellent. Reliability, when technically distinguished from 'quality' refers to the expected span of time that a system or device will meet the user's need before 'failing'. The term 'failing' means, exactly how the intervals of good services are defined to arrive at a mean value. For example,

Received 28 October 1996

in a redundant software-controlled system, a software bug may produce a few inappropriate operations, leading to program interrupt and software recovery action. In this case, the actual output of the system may not be affected. Here, the statistics of the software reliability are affected, but the system down-time may not be affected.

2. RELATIONSHIP BETWEEN HARDWARE & SOFTWARE

There are many possible distinctions between hardware and software. These differences arise from the fact that a software is essentially an abstraction and therefore is immune to any mechanical defect, electrical noise, and physical degradation that afflict hardware. Many people concerned with software engineering have pointed out that the cost of removing quality problems (normally called 'bugs') grows exponentially as time passes from one stage to another. The same is true for hardware. However, hardware also suffers from faults during manufacture or those arising spontaneously while in use due to stress or aging. These later problems have become prominent over decades, and therefore hardware quality control has been concentrated on manufacturing control rather than on design. On the other hand, quality problems in software are primarily design problems. The methods used to control the quality of software are through some 'hygienic' approaches to the design process. This is because software problems are primarily logic related that has no analogous problem of wearout. On the other hand, logical complexities from hardware are largely eliminated with the evolution of reliable, low cost memory devices and by the incorporation of standard integrated circuits (ICs) from which logical problems are eventually eliminated.

3. MICROELECTRONIC DEVICE RELIABILITY

It has been estimated that around 65 per cent of all aircraft down-time is associated with avionics. An interesting thing is that 90 per cent of these avionics system's components is of microelectronics type. From the above two statements, one can be sure that the avionics systems reliability is mostly dependent on the microelectronics component reliability. It has been identified that most of the failures are environment related. These environment-related problems are more severe in military community. There are three major types of stresses that can bring failure in a microelectronic device. These are environmental stress, electrical stress and radiation stress. The environmental stress includes: temperature, pressure, humidity, corrosive atmosphere, vibration, acceleration, etc. The electrical stress includes: excess voltage, excess current, electrostatic discharge effect, etc. The radiation stress ranges between low level cosmic radiation from trace impurities and packages to that resulted from a nuclear explosion.

4. ENVIRONMENT-RELATED FAILURES

4.1 Temperature-Related Problems

It has been estimated that 50 to 60 per cent of the microelectronic device failure is due to thermal problems. This problem is growing exponentially,

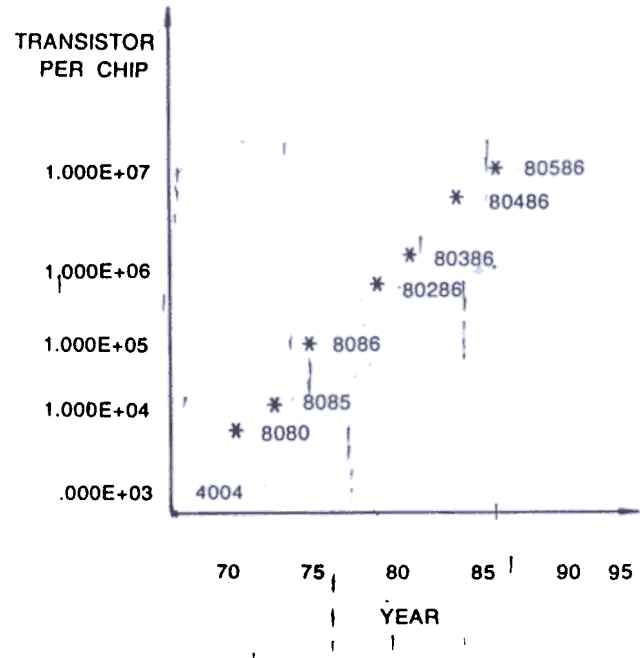


Figure 1. Power density trends

because, over the decade, the component density of the chips has increased at a faster rate. Figure 1 shows the transistor density of several well-known microprocessors. The transistor count has increased at tenfold rate over six years, while the component power density has doubled every three years. On the other hand, the environment temperature in a military environment can vary from -54 °C to +71 °C. For an avionics application, this rate of change of temperature is very high and can lead to two types of detrimental effects: (i) at high temperature environment, the power dissipation capability of the device package decreases leading to thermal breakdown, (ii) due to rapid temperature changes, cracks may develop in the package, which will lead to the absorption of moisture and subsequently to the development of corrosion processes, increase in leakage current, etc. This is also true in the case of plastic packaged devices, because of the differences in the thermal expansion coefficients of the package and silicon substrate.

4.2 Pressure-Related Problems

Low pressure reduces the dielectric strength of the microelectronic component and air density. This may not produce complete electrical breakdown, but corona and its undesirable side effects stimulate the situation that may increase the losses in microelectronic components. Besides the ionisation effect also takes place. As the air density reduces, the heat conduction from the component also gets affected. Besides thermal, humidity and pressure stresses, the mechanical stresses, such as vibration, acceleration and shock, etc can also create problems in the device package. These stresses can cause lead breakage which is reflected in their mechanical integrity, electrical characteristics, or both. Accurate modelling of potentially destructive mechanical and thermal stresses offer enhanced electrical reliability for both ICs and packages. By devicing models to simulate the effects of stress on critical structural elements of a device, manufacturing processes can be designed to reduce stress at sensitive points and thus minimise IC failures.

Considerable efforts have been made to reduce the susceptibility of IC packages to the environmental stresses. One of the recent approach is by finite element analysis (FEA). Using this technique as a computer-aided design tool, a semiconductor package design engineer can characterise stresses and deformations throughout the package for any type of mechanical or thermal load. This method demonstrates exactly where the excessive stresses occur in an IC package and why? For example, shear stress parallel to the chip surface reaches a maximum at the corners and decreases to zero at the center of the chip. Similarly, an increase in the thickness of the package to increase the structural strength will have an impact on the thermal characteristics of the chip. Finite element analysis modelling can also determine how tight the assembly and packaging process must be, to increase the package reliability.

5. RADIATION EFFECTS

Several types of radiation are of primary concern to military and aerospace applications. Ionizing radiation create electron-hole pairs as they pass through a material. It can also create defects in the silicon crystalline structure. The recently discovered radiation effect on semiconductor is the single event upset. This is associated with heavy, high energy ions that create very wide and deep electron pair tracks with potential energy, high enough to generate voltage spikes throughout the circuit. The main sources of this kind of radiation are solar flares, alpha particles (from radioactive contaminants in electronic packages) and neutrons and protons (from the nuclear reactors). In general, digital devices are mostly unaffected, until the radiation effects exceed the threshold levels. A linear device responds quicker but may survive longer.

To date, silicon on sapphire (SOS), silicon on insulator (SOI) and dielectric isolations have been the best choice for the radiation environment. These techniques basically reduce the junction volume available for the electron-hole pairs generation in bulk silicon. Moreover, carriers

generated in supporting silicon substrate are not collected by the device nodes, because the devices are electrically isolated. The dielectric isolation also eliminates parasitic field devices, which improve the circuit's total gama doze hardness.

6. ELECTRICAL ENVIRONMENT

On one hand, working energy levels of today's electronic systems are becoming low, enabling the integration of lakh of transistors inside a chip. On the other hand, devices are becoming more and more prone to environment like static electricity, spikes, transients, etc. Electrostatic voltage may cause junction burnout in bipolar chips and dielectric breakdown in MOS and CMOS devices. Spikes and transients may cause unpredictable logic states in digital systems. The use of antistatic materials, surge protection circuits, etc are some of the techniques used to protect against these environment.

7. AREAS OF QUALITY & RELIABILITY ASSURANCE

The source of a system failure starts right from the system requirement specification, hardware failure, hardware design errors, software coding errors, software design errors, and human errors. In a complex system, failure can occur due to unusual combination of any of these. This means that the efforts for quality and reliability must start from the early phases of product development to be continued till the field deployment of the product.

7.1 Applicability to Hardware

At hardware level, one obvious technique is to use very reliable components. The MIL-SPEC program codifies standards for many kinds of devices that the military procures. The MIL-HDBK- 217E[1] gives the reliability model to estimate the failure rates for various kinds of integrated circuit chips. The MIL-STD- 883E[2] gives the screening method and procedures to make a reliable microelectronic device. Here, the environment is categorised as:

(a) Physical environment, in which components are exposed to various physical environment such as temperature, low pressure, humidity, etc.

(b) Mechanical test, in which the mechanical integrity of the basic chip with the package is verified. The device will be exposed to vibration, acceleration, shock, etc and the lead integrity is verified. The visual examination is also conducted to identify other mechanical integrity problems.

(c) Electrical stress, in which the device is subjected to various static and dynamic electrical stresses. All functional tests are carried out in this stressed environment.

Above the chip level, techniques for building reliable devices include: component burn-in, careful signal routing, shielding, cabinet grounding, environmental control, and other conservative and well-established design practices. In a very large system, it is unreasonable to expect that every component will be totally reliable. In this case, other techniques that allow for individual component failure must be used. For this reason, a number of techniques have to be implemented that allow a system to continue functioning even when individual components fail. These techniques involve redundancy management and dynamically configurable systems. The redundancy techniques that allow for individual component failures, themselves add additional complexity and possible sources of error into the system.

Another problem area in hardware is the environment in which the system is working. Figure 2 shows the percentage failure rates due to various environments. Two methods are used in military community to build confidence in the system against the environmental problems. One is the environmental qualification testing, in which the prototype of the system is subjected to simulated environment before production to prevent these common problems in an equipment that must operate at high reliability in severe environment. Military applications development engineers use a variety of equipment for environmental stressing. They expose prototypes to

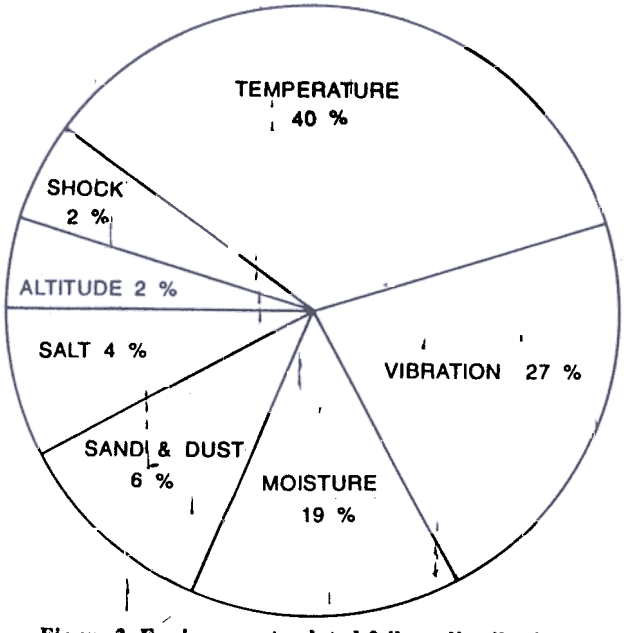


Figure 2. Environment-related failure distribution

extremes of temperature, altitude, and humidity in specially built chambers. Shakers, shock machines and centrifuges are used for dynamic stressing[3]. The second one is the environmental stress screening (ESS) which can be applied to both prototype design and actual production units. If the ESS program is started at the initial stages, defects in the product design can be corrected before the production starts[4]. The ESS identifies most of these defects at the factory, reducing manufacturing costs for quality control and rework, as well as lowering the warranty and maintenance costs for both the manufacturer and the user. It can also lead to changes in the design itself, based on the result of screening at the production units (Fig. 3).

The difference between qualification testing and ESS however is significant. The qualification testing process tests a product to uncover defects in design and does not require every prototype to be tested because the same design flaws will occur in each protype of the same design that is produced. The ESS on the other hand requires that every single unit produced be screened for defects that have occurred primarily during manufacturing. These flaws may vary from unit to unit. The effective implementation of ESS depends on the

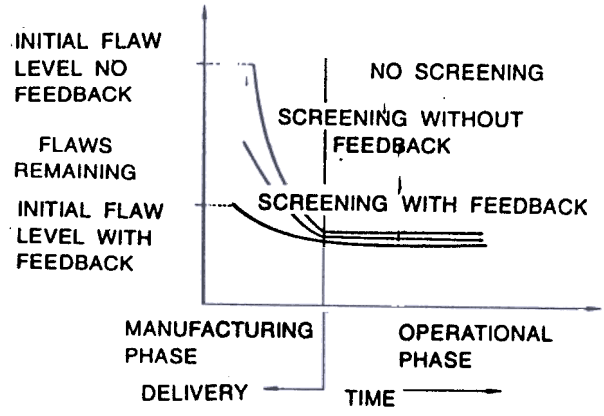
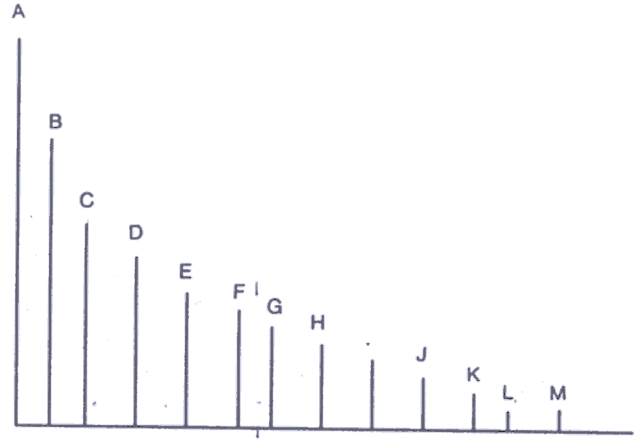


Figure 3. Effectiveness of ESS on bath tub curve



A - TEMPERATURE CYCLING
B - RANDOM VIBRATION
C - HIGH TEMPERATURE
D - ELECTRICAL STRESSES
E - THERMAL SHOCK
F - SINE VIBRATION FIXED FREQUENCY
G - LOW TEMPERATURE
H - SINE VIBRATION SWEEP FREQUENCY
I - COMBINED ENVIRONMENT
J - MECHANICAL SHOCK
K - HUMIDITY
L - ACCELERATION
M - ALTITUDE

Figure 4. Weighted rank effectiveness of ESS

identification of the environment which cause maximum infant mortality failure in the system. Figure 4 shows the weighted rank of effectiveness of typical environmental screens (data from

Institute of Environmental Science). It is evident from the figure that most of the failures are revealed with temperature cycling and random vibration. That is why, ESS has got another name, i.e., 'shake and bake' for reliability.

Another important area having great implications is the electrical environment in which the system is working. Today, the electronics are more vulnerable to electrical interference, such as lightning, electrostatic discharge, electromagnetic interference, electromagnetic pulses created by nuclear explosions, etc. As the system complexity grows, it is necessary to incorporate some online test strategies into the system. This is evident from the functional capabilities of today's microelectronic devices and systems. In a complex redundant system, isolation of the system faults and reconfiguration of the system are necessary after a component or sub-system fails in the system. Ideally, test strategy should be mapped out at the time of product conception (Fig. 5). This test strategy will vary greatly from product to product, depending upon the system requirements. Also, in an ideal environment, test engineers work side by side with design engineers, software programmers, and hardware design engineers. This permits test

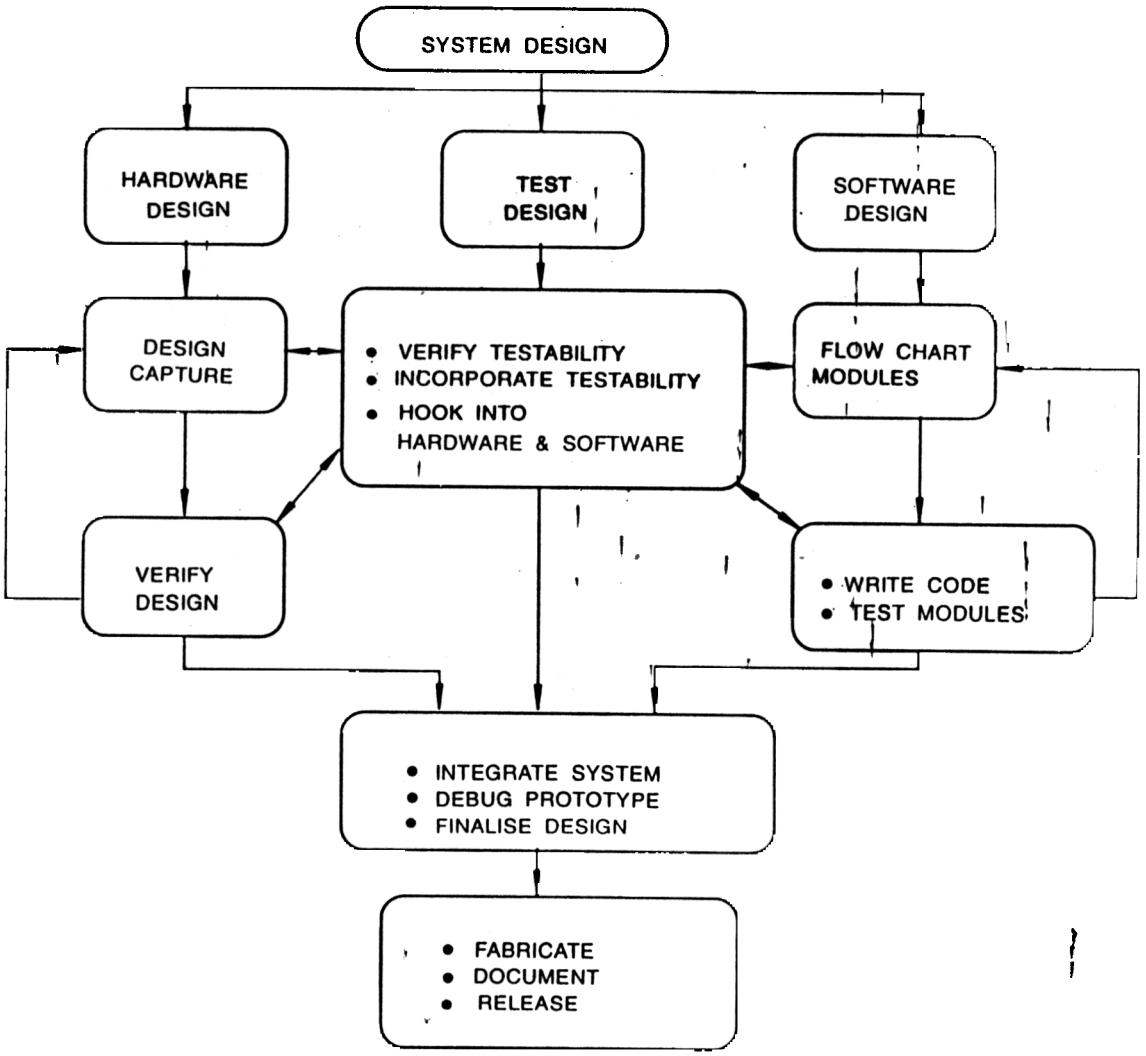


Figure 5. Design for test

features to be incorporated at the time of implementation, rather than adding components later, that affects other system features.

There are different approaches to design for testability. Built-in test (BIT), Built-in test equipment (BITE), and Built-in self test (BIST) are some of the techniques. Implementation of all these depend on the complexity of the system and some other requirements, such as system efficiency, fault tolerance, etc. The BIT refers to a component or assembly having built-in test capability but that requires some external source to actually analyse where faults occur. The BITE refers to a system having a section dedicated to incorporating test equipment that can exercise system to locate faults. The BIST refers to a component or assembly having circuitry built-in that can actually test itself with proper pattern sequence exercising it.

7.2 Applicability to Software

For large computer systems, the cost and complexity of the software typically dominate hardware. The software programs contain trade-offs between economy of memory and speed of execution. For example, execution time is lost when transfers are executed to reuse code, but at the cost of memory, straight line coding could be used at higher speed. So in any software system, there will definitely be a trade-off between hardware and software.

Software quality is adequately different from hardware quality that it warrants separate treatment. The quality problems in software are primarily design problems. Software design is the creation of detailed description of a logical process which is perplexing because the evolving products continually compel engineers to reset directions throughout the project. Software designers are often called redesigners because they frequently make improvements in their design whenever they find a better way.

The lack of quality and consistency in delivered software products is due to lack of methods, visibility, consistency and resolution of ambiguities in system and software requirements. So the basic approach in any software design is to find and eliminate the errors as early as possible in the software life cycle, the by-product of this approach being improved quality, higher productivity, and less time wastage during testing. The experience has shown that the use of high order languages in coding, once exclusively used for data processing domain, became acceptable, and infact are being preferred for embedded systems, given the adequacy of processor performance. The computer-aided software engineering (CASE) tools has become very useful during all phases of software development cycles.

In today's electronic systems, in addition to carrying out the main functional operations for redundant systems, the software also play a key role in maintaining system integrity in the face of hardware malfunctioning. In such systems, software failures can be categorised into the following:

* Failure to perfom functions as required under normal conditions

  Failure to perfom as expected under abnormal conditions

  Failure to recover after a hardware-related failure and

* Failure to diagnose hardware failure

Today's electronic systems have become complex. After considering the abovementioned failure criteria also, the design cannot be said to have adequate quality unless its performance is satisfactory in the face of all expected conditions. A useful measure of software quality is the mean-time between failures (MTBF). Since failure come with varying levels of importance, it is necessary to grade them into several levels ranging from fatal flaws to cosmetic problems. Also, one needs to relate the density of bugs to the likely rate at which bugs will create actual failures in field conditions. Although software reliability can be assessed by testing, the mean-time between failure statistics is very sensitive function of usage. Thus, if the test sequences are imperfect in the sense of

actual field usage, the results are not likely to accurately reflect reliability. However, experience has shown that bug density is a useful predictor of reliability.

7.3 Software Quality Control

The essential elements of software quality control are: (a) Means of measuring conformance to requirements and (b) management strategy to assure corrective action for ensuring that conformance to requirements is maintained. By following a well-established design methodology, a well-designed test plan and a well-established configuration management plan, it is possible to assure extremely reliable software. To ensure a quality software, the 'bug density' should be tracked and an acceptable level should be established (for complex system application one bug per thousand line is an acceptable limit). The system test must include exposure to a stressful environment without evidence of fatal flaws. An ideal condition for testing is impossible to be achieved. All software designers are familiar with the sequential approach to debugging in which modules are tested until they meet module specification, and then tested modules are integrated into a complete software for final test. The final test is more stressful than the module test because a new set of problems may emerge. There are two methods to tackle this problem. One is the check for test coverage. Test coverage is actually a measure of the proportion of the program actually exercised during a test. Branch points of the code are tracked during the test to see which portions of the code are tracked and also to see what proportion of the alternative paths have been taken. The second method is overload. It is intuitively clear that problems are particularly expected to occur under stresses which occur when registers go into overflow, stack overflows, more number of interrupts, etc. Along with software maintenance, simultaneous software updates are the added stresses. Software which work well under all these conditions can be expected to be free from bugs. As stated earlier, the reliability cannot alone be dependent on the test strategy. This is especially true in the case of complex systems. Here, it is necessary that a well-designed method for a project control should be invoked with the assumption that the problems are avoided right from the earlier phases of the development. The importance of this aspect of quality assurance is evident from the rapid cost escalation as problems remain undiscovered while development proceeds. No one can claim that his method is 100 per cent foolproof. What is needed is a review process to assure that design does not become entrenched before requirements are thoroughly clear, coding does not begin before the design is settled, and the product is not delivered before the stress testing mentioned above has been completed.

8. IMPLEMENTING SOFTWARE QUALITY ASSURANCE FUNCTIONS

The software quality assurance (SQA) is a planned and systematic pattern of all actions. The minimum subsets of a software quality assurance function are[5-10]:

(a) Management

(b) Documentation

(c) Standards, practices and conventions

(d) Testing, life cycle audits, audit checklists, and

(e) Configuration management.

8.1 Management

The main problems associated with the failure of a software product are inadequate planning, and inadequate requirement specification. The planning activity is mainly based on the project to be executed, the customer's requirements, and the experience of the manager. The output of the

activity is a series of documents, project management plans, schedules, cost planning, computer program development plans, test-plans, etc. However, an ideal model for the process is that of software life cycle model, which projects that a good requirement specification will exist by the end of a specific phase. Projection of an inadequate requirement specification is perhaps the most serious problem in software development. It is at this level, that the systems connection with the outside world is expressed. Human factor plays a very important role here. We have the inability to grasp in totality, and also, further inability to communicate what we have grasped. This is reflected into the specifications that lack completeness, clarity and consistency. The second thing is that the user himself may not be fully aware of the things he really wants. To solve this, the trend appears to be evolving towards:

* Recognising a way of life that requirements will confirm to be incomplete.
* Encouraging schedules that explicitly recognise incomplete requirements.
* Promoting early identification of requirement specifications changes and correct disposition.

  Ensuring that software is designed to enhance change implementation.

8.2 Documentation

As compared with hardware for which both the document and the product are visible throughout the product life cycle, the visible element of a software product is only documentation, without which there is neither process nor product. Apart from this, there are more concrete reasons for documentation.

(a) Writing down decisions is essential; only when documented, gaps and inconsistencies appear and decisions already made come into clear focus.

(b) Documentation communicates decisions to others.

(c) Documentation offers database and checklists for periodic reviews.

(d) Documentation offers a clearly definable position, essential to demonstrate to the customer, how requirements and specifications are met.

Table 1. Document set recommendations

Document class	IEEE	DOD 2167	DOD 2167A	NASA
Management		4		25
Engineering	2	9	7	21
Testing	2	4	3	6
Support	1	7	5	4

Table 1 shows document set recommendations for software development activities in the US. A review of the efforts made by the three organisations reveals that there is yet no clear answer to how much documentation is enough. Instead, each standard recommends that documentation be tailored to fit a project need. The IEEE standard recommends the possibility of addition to the minimum set while DOD and NASA recommend the possibility of subtraction from the maximum set.

8.3 Standards, Practices & Conventions

The main difficulties with the standards have not been their existence, but their recommendations and ability to be implemented. The ability of a standard to be implemented depends on two factors.

(a) The environment in which the standards are embedded and

(b) The ability to determine objectives if standards are being followed.

Both these factors are inclining more in the direction of implementation into an overall programming environment in which the standards, practices and conventions are human-factored into the software supporting the development team.

8.4 Testing, Life Cycle Reviews, Audit Check List

The software testing is recommended as a primary tool for the software quality assurance. Testing means not just running the tests, but designing tests, establishing test standards, designing correction procedures for discovered errors, etc.[11] The activity required to create a test is a powerful bug preventer but the problem is that the test design actually begins only after the first development stage and at this stage, enough product details are available to allow the drafting of test design. Another problem is that the tests must be selected from an infinite number of input, output values and paths which may lead to errors in testing. Normally three types of errors are created during a test design:

* Creating too few tests, letting too many bugs get into the customer.

  Creating wrong tests, letting the wrong bug get in to the customer.

  Creating too many tests, doing unnecessary work.

To overcome such errors, the tests must be comprehensive and should include the following:

- The test must match as closely to the test environment as that of the customer's environment.

  Get customer's participation during testing.

  Design the test against a history of customers reported failures on previous product.

  Historical failure information should be recorded to create a database.

Life cycle reviews and audit checklists are considered the most powerful bug preventer in contrast to testing. This is because reviews and checklists run the entire length of the product cycle. Reviews are dynamic as they deal with the on-going development of the software system. They are also preventive, because they are intended to stop problems before these appear as a bug in software product. Audits, in contrast, are designed to check the state of the system being developed at a certain point. They also assure that the current product is in accordance with the organisation policies, procedures and all system requirements. Audits are static as they deal with the past development of the system that leads to the current

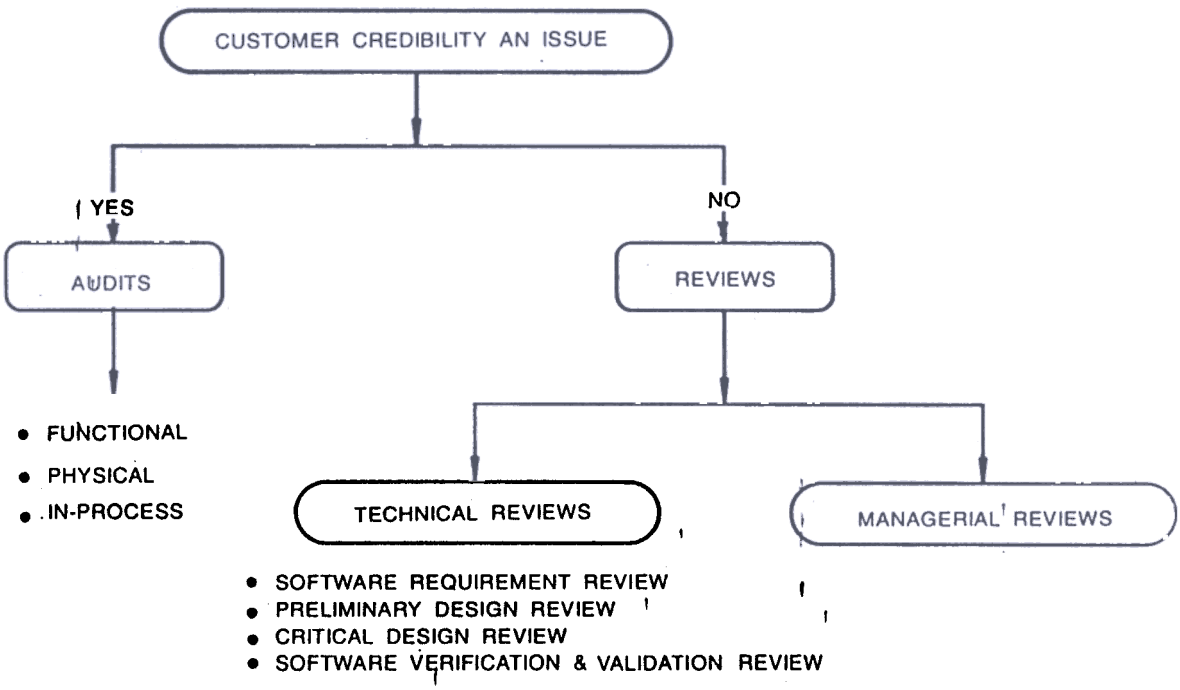


Figure 6. Review techniques decision tree

software system product and corrective, because these aim at correcting faults present in the supporting materials (Fig. 6).

8.5 Configuration Management

The effective implementations of configuration management vary in depth according to the project parameters. This is normally combined with code control, media control and problem reporting and corrective action as a discipline. The principle management tool of software configuration management is the 'baseline', a formal milestone in the product system life cycle, usually defined at the end of each phase of the software life cycle. The baselines are some documentation representing the approved configuration identification of the program against which the software product under development is compared to ensure that it conforms to the design specification in the baseline documentation. The following points can be considered for SQA recognising a series of functions to be tailored to fit particular organisation :

(a) Quality is designed into software, not tested in

(b) SQA should be involved right from the beginning and should participate in each stage of the project.

(c) SQA should be central and independent.

(d) The minimum efforts for SQA should include

* Establishing a library of standards, procedures and technical publications.
* Reviewing document for completeness and conformance
* Establishing change control procedures

(e). Mid-level efforts for SQA should include

* Establishing configuration management procedures
* Participating in the development of document, reviews, walkthroughs, and audits.

(f). High level efforts for SQA should include

* Preparing comprehensive test plans
* Maintaining a baseline library for all project documentation and code
* Maintaining a test library of test plans, test reports, and evaluation of test techniques
* Maintaining concurrency with a state-of-the-art analysis, design, programming and testing techniques.

9. CONCLUSION

The challenges to continue improvement of quality and reliability come from two primary trends: advances in hardware and software technologies. Hardware technology has advanced by increasing scales of integration. Functional capabilities of microelectronic devices have increased. Inspections have become difficult. Due to low energy levels, the errors may be caused by stray-charged particles. Along with these new technologies, it is essential that the quality assurance technologies keep pace with appropriate ways of monitoring processes, predicting reliability and carrying out failure mode analysis.

The software technology produces codes in enormous quantities. A one million line program may contain thousands of bugs. The challenges to software programmers are through better design, effective development tools, sophisticated test methods and improved management of development processes. Improving the quality is the only way of improving productivity. High technologies that create challenges must support the need of quality and reliability improvement. Through the use of sophisticated quality assurance techniques, we can look forward to the day when quality analysis will celebrate the discovery of each new defective part or line code as valuable input to the improvement process.

REFERENCES

Military handbook on reliability prediction of electronic equipment. US Dept. of Defence. Washington, DC. MIL-HDBK-217E.

2. Military standard on test methods and procedures for microelectronics, US Dept. of Defence. Washington, DC. MIL-STD-803E.

4. Military standard on engineering stress screening process for electronic equipment, Dept. of Defence. Washington, DC. MIL-STD-2164.

5. IEEE standard for software quality assurance plans, ANSI/IEEE, New York, 10017. ANSI/IEEE STD. 730-1984.

6. Defence system software development, US Dept. of Defence. Washington, DC. DOD-STD-2167A.

7. Software considerations in airborne systems and equipment certification. Radio Technical Commission for Aeronautics (RTCA), Washington, DC. RTCA-DO-178B-1992.

8. Buckley, Flether J., & Robert, Poten. Software quality assurance. *IEEE Tran. Software Engineering*, 1984, **SE-10**, (1), 36-41.

9. Alan, Boming. Computer system reliability and nuclear war. *Communication of the ACM*, 1987, 30(2), 112-29.

10. James, Vincent; Albert, Waters & John, Sinclair. Software quality assurance, Volume 1. Prentice Hall, Englewood Cliffs, New Jersy, 1988.

11. Richard, Hamlet. Special section on software testing. *Communication of the ACM*, 1988, 31(6), 662-67.

Contributor

Dr V Haridas is working as scientist in Regional Centre for Military Airworthiness (Aircraft), Bangalore. He is BTech in Electronics & Communication Engineering. He completed one-year Electronics Fellowship Course at IAT, Pune, in 1993. His areas of work are verification, validation, testing and evaluation of avionic systems and safety critical airborne software.