Accelerated Stress Testing for Both Hardware and Software

H Anthony Chan, University of Cape Town, Cape Town

Key Words: Stress testing, reliability

SUMMARY & CONCLUSIONS

Accelerated Stress Testing (AST) has been used in electronic, electromechanical, and mechanical systems to achieve robustness with high reliability primarily for hardware. For software products, the reliability program is often conducted separate from any hardware accelerated stress testing. Yet, many systems consist of concurrent software and hardware issues.

In addition, the stress testing processes were primarily adopted by those responsible to develop and manufacture hardware. For example, the stresses usually include temperature extremes, thermal cycles, vibrations, etc. These stresses are effective in accelerating latent hardware defects from degradable, marginal, or intermittent failures to hard failures so that root cause analyses and corrective actions may be made. Although experiments had indicated that software faults and hardware defects are related, the available formulation of the fundamental principles [1] was still based on hardware systems. AST for software and for operating systems have been discussed in [3] and [4], but a fundamental understanding of AST for software is lacking.

In order to generalize the fundamentals of accelerated stress testing to address both software and hardware, we need to define accelerated stress testing for software and to address whether they are needed, i.e., whether there are effective methods to achieve high software reliability.

The basic reliability concepts categorize systems into different categories according to the presence of defects and faults and whether these weaknesses are explicit enough. The concepts for both hardware and software reliability separate the notion of defects and faults from failures. It further conceptually separates the notion of stressing and the notion of detection. The fundamental concept is that all failures except the explicit ones must be manifested under certain stress conditions. There is then a threshold stress level beyond which a system will fail. The cumulative effect of stresses is included by defining time as one type of stress.

Both hardware and software systems have marginal weakness, and degradable weakness. The process of recovery and repair are also examined for both hardware and software events.

The basic reliability principles in accelerated stress testing for both software and hardware systems are combined and explained in this paper. While [3] and [4] also address the needs and advantages of AST for software, an effective software AST program will require efficient tools yet to be developed. The benefits should justify the needed further research and development in this area.

1. *INTRODUCTION*

Many products nowadays comprise both hardware and software, both of which need effective and efficient reliability methods to achieve high reliability at low cost with short time-to-market. Accelerated Stress Testing (AST) is an effective method of achieving system robustness by detecting product weaknesses using accelerated stresses, conducting failure analysis, and taking corrective actions.

Products may often have hidden defects or weaknesses, which can result in future failures in the field. AST applies stress stimuli to a product to turn such latent defects into observable failures, and therefore offers opportunities to discover and correct product weaknesses early in the product life cycle.

The field of accelerated stress testing had undergone different modes of usage and development even when they were still considered empirical. The predecessor of AST was environmental stress screening originated in the 1960's in the space program in USA to achieve 100% defect free system. It was later used in 1970's in the defense electronics industry in USA as an effective quality control technique. In the 1980's the commercial electronics industry used it to also achieve high quality with a lower cost constrained by a shorter time-to-market. The methodologies developed at that time were often empirical. By the late 1990's, AST has been adopted by many equipment manufacturers in major global industries including those of telecommunication, computer, network, and health care to achieve robustness and high reliability primarily in hardware systems.

These unconventional methods often appear contrary to conventional methods when products are seemingly overstressed. Numerous questions and issues were raised resulting in much controversy at that time. The need to better understand these empirical methods and to address the questions had led to a systematic formulation [1] in the 1990's but the formulation was primarily discussed in terms of hardware reliability. Many systems consist of both software and hardware combined together. Generalization of AST to both software and hardware systems is therefore desired.

The emphasis of this paper is to define, clarify, and elaborate the theoretical concepts, as opposed to providing practical procedures or results. The majority of existing papers are already in the latter category. Those interested in the latter may find plenty of information in this subject elsewhere such as [2] which also contains a bibliography of over 300 references.

2. *HARDWARE VERSUS SOFTWARE RELIABILITY CONCEPTS*

The concepts of AST require careful definition or clarification of several fundamental reliability concepts. Some of these defined in [1] were primarily based on hardware failure experience. Some of them are still applicable to software reliability, but others need to be re-defined or changed. We need to carefully examine these basic reliability concepts first.

2.1 *Defects and Faults versus Failures*

A striking similarity between the fields of hardware reliability and software reliability in the development of these basic concepts is that they both have independently separated the concept of product weaknesses and faults from failures. In software, codes may have faults but these faults are not yet failures. They are seeds for potential failures. The encountering of certain other conditions is needed for the faults to manifest as failure events. The software faults are analogues to the product weaknesses in the reliability concepts in AST.

A basic reliability concept already discussed in [1] is to categorize any product units into the good, weak, and bad categories. Good products meet all design objectives under all nominal stress conditions.

Bad products have hard defects or parametric deficiencies, for which failures are observable and certain to impair the usefulness of the product. These bad products are, in principle, detectable, although they may pass the detection because practical tests may be non-ideal and often do not have 100% test coverage.

Weak products have defects and/or faults that are commonly not immediately observable, and may exhibit a static or a degradable nature which we discuss later in Section 2.4.

The above categorization has distinguished the concept of defects and faults from that of failures. For example, a typical test of a product may produce only a binary output: pass or fail. It is then tempting to assume that an ideal test has successfully separated out the defective product units by associating the defective ones only with those that failed the test. Indeed, if products have only good and bad categories, then with ideal tests that have 100% test coverage, the test results will be deterministically pass or fail. We will see later that with the addition of weak products, the test results will no longer be deterministic.

Defects and faults are attributes of a product and may be present regardless of whether or not the product ever exhibits failures, which may occur only under certain test or use environments.

Tables 1 and 2 show the defects and faults as attributes of the product in the rows and the pass and failure as outcomes of detection and field use.

2.2 *Stressing versus Detection*

The earlier reliability concepts in the AST formulation [1] further clarify the role of testing by conceptually distinguishing between stressing and detection. Product units in the bad category are defined as those that will exhibit observable failures when subjected to a failure detection process with 100% test coverage.

Table 1. Defects versus failures with non-ideal detection.

	Detection with insufficient test coverage	Field use
Good: has no defects or faults	pass	No failure
Weak: has soft defects or dormant faults	pass	May fail
Bad: has hard defects or observable faults	Pass or fail	Increased probability of failure

Table 2. Defects versus failures with ideal detection.

	Detection with 100% coverage	Field use
Good: has no defects or faults	Pass	No failure
Weak: has soft defects or dormant faults	Pass	May fail
Bad: has hard defects or observable faults	Fail	N/A (not shipped to customers)

Strictly speaking the test coverage here is "detection" coverage, because many functional tests are not intentionally conducted in conjunction with stresses. In software reliability, the certain conditions under which failure events occur are analogous to the appropriate stress conditions under which hardware failures take place.

We note that the challenging task of improving the test coverage, generally in the restricted sense of detection, is itself an important area. Yet 100% detection coverage alone does not get rid of field failures. The improvement of a 100% detection from an non-ideal detection is seen by comparing Tables 1 and 2. Yet, a successful AST program does require good detection methods with high test coverage, this however, is beyond the scope of this paper.

We also note that the separation of stresses from detection may be conceptual because most tests are usually accompanied with some stresses. For example, testing whether an incandescent light bulb is functional by turning it on has already subjected the light bulb to a stressful transient current during the turn-on process.

2.3 *Threshold Stress and Time Stress*

In [1], 3 types of failure modes were defined: threshold-stress failure, cumulative-stress failure, and combined threshold – cumulative stress failure. Also, the concept of life-time maximum stress for one particular system was introduced there. (1) Threshold-stress failure is defined as the largest peak stress or combination of stresses, which are encountered by that system throughout its entire product life.

Whether the system fails during its product life will therefore depend on whether it can withstand this maximum stress level. (2) For cumulative-stress failure, we note that time may be included as a stress. The combination of time-stress and other stresses are what manifest the cumulative-stress failure. This combination also has a maximum time-stresses combination over the product life of any system, and therefore also has a threshold value to manifest failure. (3) Combined threshold-cumulative stress failure is manifested by a threshold-stress first followed by a cumulative-stress, or vice versa. An example of such a software failure is the overflow of an interim event counter that occurs because a low priority reader process is indefinitely postponed by other high priority processes. Raising the priority of the reader process raises the threshold at which the stress can begin to accumulate. Increasing the size of the event counter increases the amount of continuous stress that can accumulate before the overflow occurs.

2.4 *Weak or Marginal*

The results of testing for defects and faults are not necessarily deterministic. Instead, the system may just have a certain probability to exhibit failure. This probability is often dependent on the stress conditions, which includes the types and magnitudes of stresses or combination of stresses applied to the system. These systems are weak or marginal.

An important attribute of the weak or marginal systems is whether they are degradable or recoverable.

2.4.1 *Degradable*

The probability of failure for a system with defects or faults may remain constant or may change over time. We may then refer to the system as being degradable or not.

A failure event may not be accompanied or preceded by degradation. Malfunction or failure may occur whenever a specific action is encountered, without being preceded by a degradation process. The probability to fail for these weak products may be small in the absence of certain stresses but may increase under certain stress conditions. In hardware, these weaknesses could exhibit marginal behavior.

Examples for software are the execution of a faulty function, and the dysfunctional processing of unanticipated data, e.g. as when some subroutine neglects to perform a boundary check prior to dereferencing a null pointer.

Degradable products have latent defects and may degrade irreversibly into bad or marginal products. The degradation may be stimulated or accelerated by certain stresses. The product strength deteriorates irreversibly under stress until failure is manifested. After the product has been degraded, removal of the stresses will not restore the system to its original un-weakened state.

The concept of degradation cannot apply directly to the context of software codes, because the codes are rarely changed by executing them. (Exceptions are that some codes are self-modifying, but it is more rare now that memory is plentiful. There is also the case of corruption, either accidental or intentional). What changes however is the object system comprising the methods and data. The weak object may degrade in performance or functionality as the object is used and stressed. Examples are faults causing memory leak, memory fragmentation, and non-terminated thread.

2.4.2 *Recoverable*

In some cases it may be possible to take action that will restore the product to its original state. For example, a computer system with memory leak problem may recover its memory space by rebooting the computer; the liquid crystal display of a watch, which malfunctions when cooled to freezing temperature will operate again when warmed back up to room temperature. A system may fail under certain stress conditions. Then the system may restore to its original state when the stress conditions are removed, and sometimes also after a system reset. In such a recoverable system, the stress-to-failure process is reversible.

2.5 *Repair and Hard Recovery*

A hardware failure may be repairable or non-repairable. For electronic components, most of them are non-repairable. In electronics systems, the repairing processes are usually achieved by replacing defective sub-systems or components. After repair, the system may be considered not the same as a new system. For example, a refurbished computer probably will sell for less than a new computer.

The context of "repairing" or fixing failures is different for software. Many software problems occur only after being stressed over extended periods. Let us examine what a user would do in one software failure scenario with a personal computer.

Although a PC may malfunction the very first time when it is tested, these catastrophic failures should have been fixed before deployment. It is then after running for some time or having many programs running together that the computer may then begin to run wild. At this time, the user would probably abort the programs. If the computer freezes, the user would then reboot the machine. In still a more serious scenario, one may need to re-install the program or even reformat the hard disk to re-install the operating system and all the software needed. In this sense, the computer is restored to its original state with the working software.

The above scenario of software failure is somewhat analogous to the hardware repair process, although the process of terminating and re-starting is often broadly called recovery by users. Yet such a recovery is not a soft recovery. The original processes together with the data are usually lost. What restarted is really a new process, although using the same software.

There is a difference between hardware repair and software repair though. A defective component of a system causing system failure may be replaced with a good component. Repair of a program may involve reloading the software. However the reloaded software is usually the same software and therefore still has the same defect as before. To fix a software bug, the vendor usually needs to do it as a patch or put it in the next release. This later process of fixing is not analogous to a hardware repair, but rather to the corrective action. (As an example of hardware corrective action, consider a design that results in a temperature sensitive circuit that may malfunction at temperature extremes. Improving the design with temperature correction circuitry is a corrective action that will prevent future failures in future production, whereas replacing defective components in the circuit in one failed product is repair.)

Table 3. Comparison of repair from hardware versus software failures.

	Repair methods	Repaired system
Hardware failures	Replace defective parts	Different from before
Software failures	Re-start or reload	Same as new

3. *BASIC STRESS TESTING PRINCIPLE*

With the above clarifications in reliability concepts, the basic stress testing principle may start with product weaknesses or faults. The AST formulation distinguishes between weak and bad product units. Yet the defective bad product units are in principle detectable so that the goal of detecting them can be achieved through perfecting the test coverage. With ideal test coverage, these product units would not be shipped to customers and therefore not be susceptible to customer field failures. The defects in weak products are however not detectable in the absence of stresses. They are the ones potentially sold to customers, who may stress them over the product life. They are therefore the systems susceptible to field failures experienced by customers.

The product weaknesses of reliability concern to the customers for both software and hardware systems are therefore not those leading to catastrophic and detectable failures, but rather those with latent defects. Defects or faults, which are product weaknesses, are the necessary conditions, but not the sufficient conditions, for the subsequent manifestation as failure events under certain conditions.

3.1 *Stresses*

It is therefore a fundamental reliability concept that appropriate stressing conditions are needed to exacerbate product weaknesses to exhibit failure. This is the key assumption for AST.

Consider first a hardware system. A reliability program in the product design may try to prevent possible failures. It may then be necessary to first understand what the possible major failure mechanisms are and then design out those failure mechanism sensitivities. Yet many systems are highly complicated and there are very many possible failure mechanisms so that it may be impractical to list them all. This is especially true because technologies are changing fast and new failure mechanisms are anticipated for new technologies. Yet the product cycles are often too short. It then is difficult to exhaust all possibilities of failure modes in order to prevent each one of them throughout the design, development, and production processes.

On the other hand, we have the assumption that all field failures must occur with certain stress conditions. The stresses for hardware AST were already discussed in many places in the literature including [1]. These stresses are physical and are of thermal, mechanical, electrical, or chemical nature. They can be counted by listing the physical stresses. Therefore, the list of different stress conditions is less difficult to exhaust than the possible number of failure modes.

For software AST, numerous stresses such as load, unexpected or illegitimate data have already been included in such tests as operational profile and load testing. The applicable stresses in AST may go beyond those in the use conditions and also include (1) elevated workload, which does not need to emulate use conditions, (2) diverse, concurrent, and varying workload, (3) random workload, and (4) chaotic or unrelated workload. Besides hardware and software stresses, there are parametric stresses such as clock speed, voltage variations, electromagnetic field interferences, and others.

The stress conditions usually depend on the environment, the type of product, and the way the product is used. Examples in terms of environment are: an indoor environment with or without air-conditioning may be milder than an outside environment that may experience more several thermal cycles; a vehicular and aerospace environment may experience more vibration, shock, temperature extremes and thermal cycles; tropical and arctic environments may experience more temperature extremes; city environment may experience more chemical contamination and corrosion, etc. Examples in terms of product types are: portable electronics may experience more mechanical and thermal shock in comparison with desktop electronics. Examples in terms of usage are: some letters in a keyboard may be used much more often than others. A public phone may be more likely subject to mechanical abuse from frustrated callers than a home phone. A vehicle with lower mileage but for short trips in city traffic may have experienced more stress than a vehicle used primarily on a highway. A computer used in conjunction with many other computers in a network will be expected to experience many more different scenarios than a stand-alone computer.

3.2 *Statistical Variations*

There are different emphases in reliability concepts between hardware and software. Some are discussed below.

The statistical variations for hardware failures exist in both the lifetime maximum stresses that a given product unit will encounter and the strength or robustness of the individual product units. In the production of hardware products, there are component and manufacturing variations so the robustness of the product unit against any single type of stress combinations has statistical variations. These variations result in a statistical variation of the robustness of the product units. Against each type of stress combinations the product robustness has statistical variations this way. With a list of different types of stress combinations, these distributions of product robustness again have statistical distributions among these different stress combinations.

On the other hand, software of the same version ought to be identical, other than errors in copying. Therefore, the statistical variations for software are not on the robustness among the different copies of the same software. There are, however, still statistical variations in the probability to failure against different types of stresses.

Table 4. Dependence of HW defects and SW faults on product robustness under fixed stress conditions.

	Product Strength
Each hardware defect	Statistical distribution among different product units in product strength against stress
Hardware defects in each unit	Different distributions among different defects
Each software fault	Identical in all different licenses of the same release
Software faults in each release	Statistical distribution among different faults in probability to fail

3.3 *How high the accelerated stresses are?*

Attempts to reproduce the effects of use stress even with acceleration would impose limits on the level of stresses. Yet in AST, the principle behind applying accelerated stresses is to increase the probability of finding product weaknesses among different product units, and to check there are significant variations in the statistical distribution of the product strength. Therefore AST does not need to simulate or reproduce the stresses in the use environment. That sets the accelerated stress level to higher bounds on the stress level.

For hardware systems, the product unit weaknesses have statistical variations. It generally would require testing a large number of such systems in order to find some failures. Even accelerating stresses of the entire life of product into a compressed time frame will not gather enough failure data. In terms of finding failures with significant statistical variations in product strengths, the maximum stress level will need to be at least several standard deviations above the mean product strength. When the sample is not large, both the mean and standard deviations will be difficult to determine because the level of confidence is poor with a small sample.

Ideally, the stress levels applied must be severe enough to precipitate all the latent defects and yet not cause damage to a good product or nucleate hardware defects that cause early wear-out and reduced life in the field for the production units. Applying such ideal stresses with ideal detection will catch all bad and weak products (Figure 1).

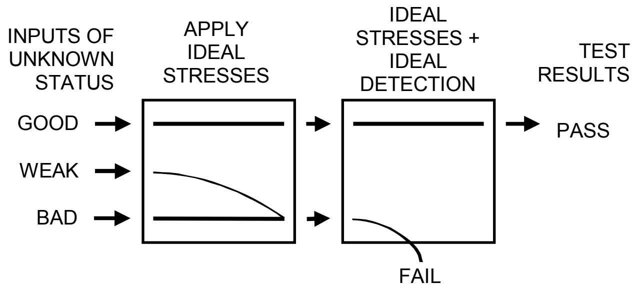

Figure 1. Schematic of subjecting product to ideal stresses and ideal detection.

Because software products do not have manufacture variations in any given release, overstressing software may only degrade at most one instance or one copy. The system may recover by either re-starting or reloading the software. There is therefore practically no limitation on the severity of the accelerated software stress. This is true provided AST is not conducted on a production system for which failures will affect customers.

Table 5. Bounds of stresses in AST.

	Example upper bound
Hardware stress	New phenomenon, new phase changes
Software stress	Practically no limits

4. *ACCELERATED LIFE TESTING (ALT) AND OPERATIONAL PROFILE TESTING*

While most tests include some stresses so that one may call them all stress testing, an important distinction of AST from other tests is that AST does not need to simulate or reproduce the stresses in the use environment.

A product or system is needed to be robust to stresses in the field throughout its product life, but not necessarily after that. It is therefore desirable to accelerate the stresses of the entire product life into a much shorter duration so that the product may be tested with these accelerated stresses through such an ALT process. Much of the effort in ALT is to find out the acceleration factor relating the ALT stresses to the real product life.

In software reliability, there are already existing names for different types of testing that go beyond functional tests to include stresses. Analogous to ALT, operational profile testing also emulates the stresses in use conditions, as opposed to creating the test profile from the functional or other point of view. Two other tests are stress testing and load testing. They include testing the response of a system upon the application of unexpected inputs. Stresses are also applied by increasing the load such as a much increased number of users and processes. They are still stresses in use condition although they are accelerated.

Such limitations on accelerating the stresses are undesired for AST. Using stresses above the ALT or operational profile stress limits enables AST to get more failures to conduct failure analysis and corrective actions. It is then up to the Failure Analysis (FA) to determine whether the obtained failures will be realistic in perspective with the in use environment. The price paid to go beyond these stress limits is that it becomes difficult or even no longer possible to find the acceleration factor to relate the AST testing time and the life of the product.

Table 6. Comparison of AST with other tests in terms of types and levels of stresses.

	Type of stresses	Level of stress	Are test failures related to field failures?
ALT	similar to field stresses	less than or equal to product life time stress	Calculate product life
Operational profile	similar to field stresses	less than or equal to product life time stress	By using field stresses
Load testing	similar to field stresses	May exceed	By using field stresses
Stress testing	similar to field stresses	May exceed	By using field stresses
AST	May be dissimilar	May far exceed	Determined through FA

5. *NEED FOR AST*

The justification for AST in hardware was originally based on success in practical cases. The need however in AST in the communication and computer manufacturers was driven by the need to achieve high reliability with lower cost and shorter time-to-market. Without the bound to relate with real life stresses, the higher stress levels give higher probability to find the latent defects.

The need for AST may be re-visited nowadays when products often include both software and hardware. Many systems are becoming increasingly complicated so that the number of possible failure modes also increases as newer products are developed. The rapid growth of high speed network has been affecting many parts of the world. A highly distributed system emerges. The conventional reliability programs cannot economically achieve high reliability.

The need to achieve high reliability under these challenges had already driven numerous software testing methods to include field stresses. However, the stresses are generally emulating those in the field.

With different types of stresses in different software testing, the number of different scenarios and the amount of testing needed have continued to increase. It is then expensive in both money and time spent to run these different tests.

Going beyond these field environment stresses opens door to much more options for accelerated stress testing.

6. *ACKNOWLEDGMENTS*

The author likes to thank Mike Werner and Jonathan Bozarth on very enlightening discussions on AST for software. Their extensive literature search and AST work on operating systems had helped the author to understand the subject. Special thanks also go to Walt Willing and Dustin Aldridge for very helpful suggestions in writing this manuscript.

REFERENCES

1. H.A. Chan, "A Formulation of Environmental Stress Testing & Screening," Proceedings of the 1994 Annual Reliability & Maintainability Symposium, Anaheim, Jan. 1994, pp. 99-104.
2. H.A. Chan and P. Englert, eds. Accelerated Stress Testing Handbook: Guide for Achieving Quality Products. IEEE Press, 2001, ISBN 0-7803-6025-7.
3. M. Werner and J. Bozarth, “Improving Customer Satisfaction via Accelerated Stress Testing of General Purpose Computer Operating Systems,” submitted to IEEE Transactions on Computers.
4. M. Werner, J. Bozarth, and H.A. Chan, “Dependability of Operating Systems using Accelerated Stress Testing,” presented at IEEE Workshop on Accelerated Stress Testing, Montreal, October 2-4, 2002.

BIOGRAPHIES

H. Anthony Chan
Department of Electrical Engineering
University of Cape Town
E-mail: h.a.chan@ieee.org

H. Anthony Chan is currently professor at University of Cape Town. He was visiting Endowed Pinson chair professor at San Jose State University during 2001-2003, and prior to that was with AT&T Bell Labs for 15 years. He has developed a comprehensive networking curriculum that combines the analytic strength from the academia with the practical experience from the industry. Anthony is Administrative Vice President of Institute of Electrical and Electronic Engineering (IEEE) Component, Packaging and Manufacturing Technology (CPMT) Society and has chaired or served numerous technical committees and conferences. He is a distinguished speaker of IEEE CPMT Society and is in the speaker list of IEEE Reliability Society since 1997.

Anthony received his PhD in Physics from the University of Maryland, College Park in 1982.