A Cohesive Framework to Quantify Computer Systems Assurance

(Presented #6011 at CMG 2006 in Reno, NV)

Dominique A. Heger
Fortuitous Technologies
Austin, Texas
dom@fortuitous.com

Phil Carinhas
Fortuitous Technologies
Austin, Texas
pac@fortuitous.com

Abstract

This study introduces a systems-engineering and evaluation methodology that focuses on the stability of an entire computing infrastructure. More specifically, the conducted research elaborates on the cohesive systems assurance (CSA) methodology, which encapsulates the concepts and methods of product assurance (reliability, availability, and maintainability), performance & scalability, and dependability (security and safety), respectively. The argument made in this study is that systems stability represents the *quality of service* provided by an entire computing infrastructure, and therefore quantifies the usefulness, trustworthiness, and effectiveness of the environment.

Introduction

This study bases the motivation for CSA on the thesis that the interrelationships among the dimensions of systems stability are paramount to the overall acceptance of a computing infrastructure by the user community. The proposed approach allows elaborating on the cohesive stability aspects of an entire infrastructure by scrutinizing and analyzing the interrelationships among the stability dimensions. This approach substantially deviates from the pervasive systems engineering and analysis process in use today, which treats the dimensions of product assurance, performance and scalability, and dependability individually in a vacuum. The proposed CSA methodology can further be utilized to quantify the relative impact that potential design alternatives may have on the overall infrastructure stability. The study first introduces the dimensions of CSA, and secondly elaborates via an actual case study on the pragmatic aspects of the methodology.

1 The CSA Equation

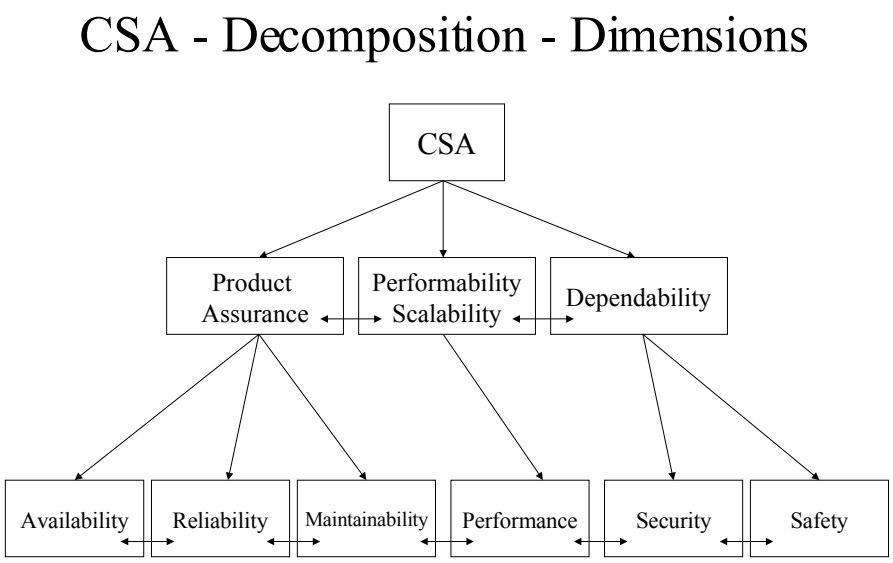
In the CSA methodology, the systems assurance dimensions product assurance, performance & scalability, and dependability are all components of the *CSA equation*. In this architecture, the product assurance component is decomposed into availability, reliability, and maintainability. The dependability component employs the security and safety issues, whereas the performance & scalability dimension discuss the traditional performance aspects of the entire infrastructure. The term *systems assurance* that is introduced in this paper is loosely based on product assurance, a term that is being used in Operations Research [10]. The argument made is that while systems assurance considerations are paramount in basically all acquisition projects, it reflects a much more involved framework for larger systems acquisitions, acquisitions of complicated and complex systems, and acquisitions of systems with significant support requirements.

The CSA equation reflects a *figure of merit* that allows identifying the component(s) that detract from the overall stability of an entire infrastructure, and ergo depict(s) the greatest potential for improvements. Each component of the CSA equation is expressed as a probability value. In the CSA methodology, the CSA equation is being used as a communication tool among the project teams. The newly proposed CSA equation is considered as a significant evolution to the *effectiveness equation* as described by Blanchard [3], Pecht [11], Landers [10], or Barringer [2] et al. The argument made is that the effectiveness equation only considers the availability, reliability, maintainability, and capability components, and therefore omits any potential scalability, security, and safety issues that may have a profound impact on the overall stability of an entire computing environment. To reiterate, the CSA equation proposed in this study is comprised of:

- *CSA stability = availability * reliability * maintainability * security * safety * performability*

Figure 1: CSA Architecture

be taken into consideration when conducting a stability analysis. To reiterate, the thesis made in this study is that CSA engineering is not described as the *quest for perfection,* but rather the search for effective business solutions that provide a very high level of quality of service to the user community.



In some circumstances, only a subset of the discussed dimensions has to be used to quantify the stability of a system. Utilizing the CSA equation leads the analyst through a very pragmatic process that incorporates analyzing all aspects of *systems stability*, by elaborating on the interrelationships among the different dimensions (see Figure 1). The CSA equation can be utilized to identify areas for improvements and to conduct cross infrastructure comparisons. The importance of quantifying the elements of the CSA equation is further tailored towards obtaining a comprehensive analysis tool that allows elaborating the relative stability aspects of any potential design alternatives. Overall, the stability of an infrastructure is expressed as a value between 0 and 1 (0% to 100%). Each element used to construct the CSA equation consists in itself of a probability value that is defined in a range between 0 and 1.

It is imperative to point out that in any computing environment, dependability and performance related issues depict an actual exercise in compromises that has to

2 CSA – Product Assurance

Product Assurance describes the extent to which a mission critical computing environment is trusted by its user community, and represents the concepts and interrelationships among availability, reliability, and maintainability [10],[11]. In other words, product assurance can be described as a (qualitative) term that revolves around the ability of a computing infrastructure to *perform appropriately*.

Reliability Dimension

The reliability *R(t1)* represents the probability that an environment operates correctly throughout a time interval *[t0,t1],* given that the infrastructure was performing correctly at time *t0*. The reliability dimension is normally expressed in terms of the mean time between failure (MTBF) for repairable entities, and the mean time to failure (MTTF) for non-repairable entities, respectively. Reliability and availability are closely related terms, but it has to be emphasized that they

do not depict interchangeable entities per se, as they represent different expressions revolving around the same issue [1]. To reiterate, reliability is focused on failure rate based statistics, whereas availability represents the measure of time a system or application component is operational [10].

Maintainability Dimension

The Maintainability *M(t)* represents the probability that a failed system component will be restored in order to resume processing within a time period *t*. The maintainability dimension represents an important component in mission critical environments where system faults might be introduced into the infrastructure during regular maintenance cycles. In other words, maintainability quantifies the *ease of repair* issue after a failure has been discovered, and concerns itself with system change scenarios such as the introduction of new application features. In most circumstances, the key *figure of merit* for quantifying the maintainability of a system is the mean time to repair (MTTR), as well as a limit for the maximum repair time [4]. Qualitatively, maintainability refers to the ease with which any software or hardware component can be restored to a functional state. The MTTR component incorporates performing corrective maintenance, which consists of fault isolation and correction procedures. Quantitatively, maintainability is expressed as another probability value.

Availability Dimension

The availability *A(t0)* represents the probability that a given system or environment is operating correctly at the instant of time *t0*, and is normally expressed in terms of the mean time between failure (MTBF) and the mean time to repair (MTTR) components, respectively [10]. As MTBF and MTTR represent the reliability and maintainability dimensions, respectively, the availability dimension depicts an actual link between the two characteristics.

3 CSA - Dependability

Security Dimension

The security aspect of an environment is considered as an infrastructure property that reflects the environment's ability to withstand accidental or deliberated attacks. Security is an essential prerequisite for availability, reliability, and safety and has a profound impact on systems and application performance [12],[13]. In terms of the stability equation, security *S(t1)* represents the probability that a system does withstand accidental or deliberated attacks in a time interval *[t0,t1]*, given the fact that the infrastructure was secured up to an agreed upon *security standard* at time *t0*. As an example, the security dimension might be expressed as the mean time to security failure (MTTSF) [13]. A less complex approach than what is being discussed in [13], revolves around assigning a certain security probability in respect to the presence or absence of certain best-practice based functional characteristics, as well as development and configuration techniques. As security aspects have a profound impact on the performability dimension, the recommendation is (if feasible) to combine the two studies into one construct. The argument made is that it is imperative to shift the industry away from only restricting access to information towards actively permitting access to different levels of information in an environment that is considered as being secured. As already discussed, the concept of security can be described as an exercise in compromises. The compromises revolve around the question of how much loss of functionality and performance an organization is able to sacrifice in order to achieve an acceptable level of security that still allows the business to operate in a very effective and efficient manner.

Safety Dimension

The safety aspect of an entire infrastructure can be described as an environmental property that reflects the system's ability to operate either normally or abnormally without the danger of causing human injury or death, or any other damage to the environment per se. Safety, availability, and reliability concerns are related issues that have to be analyzed accordingly. In terms of the stability equation, the safety *ST(t1)* represents the probability that a system does not cause any catastrophic effects in a time interval *[t0,t1]*, given that the environment was considered as being safe at time *t0*. It has to be pointed out that a system might be unreliable but safe [12]. Hence,

reliability and availability are considered as being necessary but not sufficient to guarantee safety. Reliability is concerned with conformance to a given specification, whereas safety is concerned with ensuring an environment dose not cause any damage irrespective of any conformance to a given specification.

4 CSA – Performability/Scalability

In any contemporary computing environment, the motivation for parallel processing is based on the economics of scale offered by an efficiently used parallel infrastructure [6] [7]. The argument made in this study is that any scalability model is impacted by serialization and coherency factors that have a profound impact on overall systems performance. Algorithmic constraints play a decisive role in determining the amount of parallelism, and therefore the degree of scalability that is possible for any given application environment. The computational overhead encountered in any parallel environment can be defined as the fraction of CPU capacity that can not be utilized to process a certain workload [5],[6]. This behavior diminishes the potential economics of scale offered by the already discussed parallel environment.

The argument made is that any performance study conducted in a parallel environment is impacted by the multithreaded application, the workload and its distribution, the operating system, as well as the underlying parallel hardware, and that the *performance focus* has to be on the interrelationship among the components. In regards to the stability equation, the performability/scalability *P(pt1,t1)* represents the probability that aggregate systems performance/scalability will be at or even above a certain performance/scalability level *pt1* in a time interval *[t0,t1]*, given that the environment was performing at or above a performance/scalability level *pt0* at time *t0*.

5 CSA – Methodology & Case Study

The proposed CSA architecture supports a decomposition-based methodology that follows a divide-and-conquer approach (see Figure 1). Utilizing the CSA architecture results into applying a very pragmatic analysis approach that incorporates the interrelationships and trade-off' among all the aspects of the application and systems components that compose the environment (see Figure 2).

The dimensions of CSA (or a subset thereof) are regarded as the individual input components into the CSA equation that is being used as a communication facility among the involved parties. The resulting CSA factor (a value between 0 and 1) can be utilized to:

- Quantify the relative stability of an entire computing environment
- Conduct a cross infrastructure comparison
- Evaluate design alternatives
- Identify potential areas for improvement
- Communicate relative systems stability issues and concerns in a simple, but very effective manner

Table 1 outlines the CSA data as determined on three large-scale UNIX based SMP database servers. The conducted study focused on product assurance, performability, as well as dependability, respectively. To determine the reliability, maintainability, availability, and performance factors, an analytical modeling based approach was utilized [4],[8]. Further, as the database systems were available for this study, an empirical performance and security analysis was conducted in all three environments. Quantifying the security factor was accomplished via assigning a certain security probability in respect to the presence or absence of certain best-practice based functional characteristics, as well as development and configuration techniques.

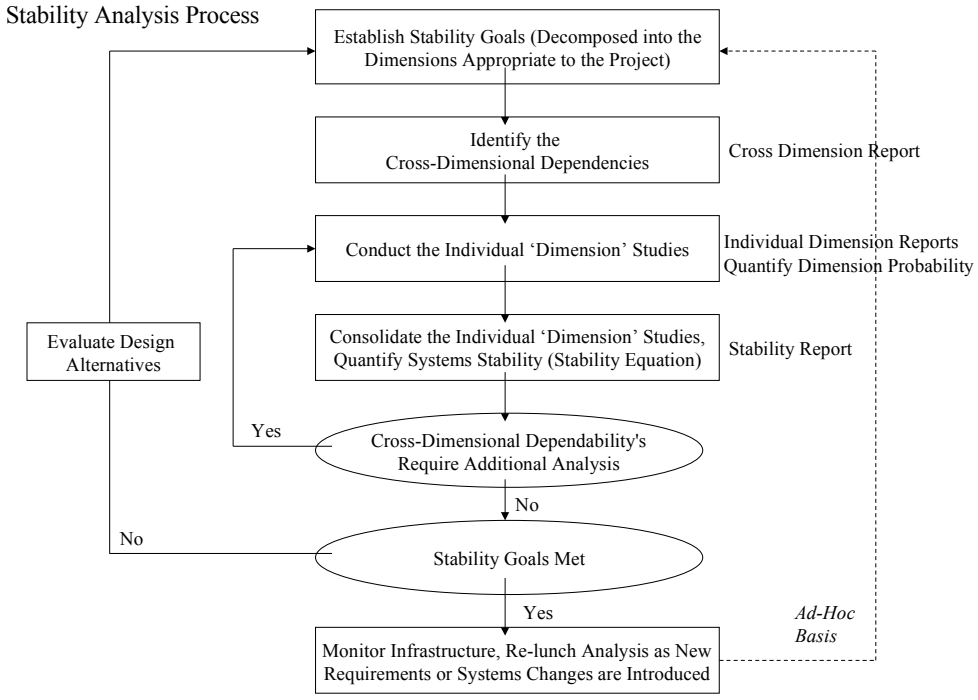
Table 1: CSA Data – SMP Database Servers

CSA	DBS-1	DBS-2	DBS-3
Reliability	0.9933	0.9923	0.9928
Maintainability	0.999	0.896	0.997
Availability	0.99997	0.99992	0.99994
Performance	0.93	0.91	0.92
Security	0.93	0.94	0.95
CSA Factor	*0.858*	*0.76*	*0.865*

The product assurance model implemented for this study combines the probability that a system will perform its required functions for the duration of a specified time interval, and that the repair action under given conditions of use is carried out within a stated time interval as.

Historical data, as well as data provided by the hardware vendors was utilized to calibrate the models. The reliability and maintainability was determined via utilizing a Weibull based analysis technique, whereas availability was determined based on MTBF and MTTR data [4],[8],[9].

Figure 2: CSA Analysis Process

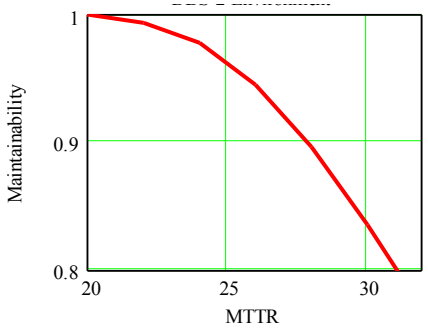


Normalizing the stability data across infrastructures has to be conducted by applying the *same rules and guidelines* for all the dimensions that are being quantified. To reiterate, the importance of quantifying elements of the CSA equation is to identify potential areas for improvements. While conducting the analysis, it is paramount to take the interrelationships among the dimensions into consideration. As an example, any security related guidelines have a profound impact on systems performance, and hence a security and performance analysis should be conducted as a unit, and should not be performed by isolating the security from the performance aspects. The same philosophy holds true for any potential availability, reliability, and performance standards that have to be met in a certain computing environment.

The data in Table 1 represents the three independent computing environments (SMP UNIX based database servers) that were being scrutinized and compared from a relative CSA perspective. The goal was to (1) identify potential areas of improvements and (2) to determine which infrastructure reveals the greatest CSA potential. The data reveals that the two systems DBS-1 and DBS-3 have an almost identical CSA value, whereas DBS-2's CSA value is substantially lower. At a first glance, all three infrastructures disclose the potential for improvements in the areas of performance and security. Further, the data in Table 1 discloses a rather low maintainability factor for DBS-2. The analysis revealed a logistical problem in that environment, and an additional Weibull simulation conducted for this study was used to communicate the availability behavior (see Figure 3). By addressing the rather low maintainability behavior, the potential of increasing the overall CSA factor for DBS-2 is striking. The model revealed that by lowering the MTTR component from 28 time-units down to 20 time-units would increase the maintainability factor to 0.9989 (from 0.896). Implementing this change would raise the overall CSA value for DBS-2 from 0.76 to 0.847, which is basically in line with the two other infrastructures. While in circumstances such as this, improving the maintainability may be the right step, the potential impact on the other CSA dimensions has to be analyzed

and quantified as part of the overall analysis (see Figure 2).

Figure 3: Maintainability Behavior – DBS-2



The thesis made in this study is that the current systems analysis process overstates systems stability, as the dimensions are quantified individually in a vacuum. In a worst case scenario, aggregate systems stability is not even being addressed. From a business perspective, the current state of systems analysis can be considered as financially inefficient, as the un-quantified systems and software stability issues have to be extensively analyzed and troubleshooted while the systems are in production.

CSA omits the danger of analyzing the different systems dimensions in a vacuum, and therefor represents a technique that approximates reality as perceived by the user community. The focus of CSA is on the interrelationships among the dimensions, resulting into a comprehensive analysis that allows quantifying the relative Quality of Service provided by the computing infrastructure. Some of the cross-dimensional dependencies that have to be analyzed and quantified include:

- Error detection and recovery mechanisms (hardware and software related)
- Security aspects such as process accounting, transaction logging, and network (stack) tunables
- High availability and fault tolerance (environmental) aspects
- Performance enhancement techniques
- Hardware and software upgrades
- Workload changes and adjustments

The methodology's decomposition based approach allows different project teams to conduct individual dimension oriented studies, but guides the analysts through a well-defined (iterative/perpetual) consolidation process. The ramification is a well understood and documented computing environment, where an organization is prepared for any contingency, resulting into an overall infrastructure that is from a financial, as well as a technical perspective, *manageable*.

6 Summary

The introduced CSA architecture, and more specifically the elements that comprise the CSA equation provide enlightenment on where from a stability perspective, ample potential is available *to be explored*. The CSA methodology introduces a synergy effect into the standard systems analysis process that benefits from evaluating the interrelationships among the different dimensions, which results into a profoundly improved infrastructure stability behavior.

The modular design of the CSA architecture allows choosing the granularity of the stability analysis on a *per project* basis. The emphasize in CSA is not on effective, but rather relative stability concerns. CSA does not pursue a quest for perfection, but rather seeks for efficient and affordable business oriented solutions that resolve the most pressing stability concerns in any particular computing environment.

References

1. *Barringer, H., Weber, D., "Where Is My Data For Making Reliability Improvements" , 4th International Conference on PP Reliability, Houston, TX, 1995*

2. *Barringer, H. Weber, P., "Life Cycle Cost Tutorial", 5th International Conference on PP Reliability, Houston, TX, 1996*

3. *Blanchard, B. S., Dinesh V., Peterson, E., "Maintainability: A Key to Effective Serviceability and Maintenance Management", Prentice-Hall, Englewood Cliffs, NJ, 1997*

4. *Gross, D., Harris, C., "Queuing Theory, Second Edition, Wiley, 1998*

5. *Heger, D., Johnson, S., Anand, M., Peloquin, M., Sullivan, M., Theurer, M., Wong. P., "An Application Centric Performance Evaluation of the Linux 2.6 Operating System", IBM Red Book White Paper, Austin, TX, 2004*

6. *Heger D., Simco G., "Quantifying the Cluster Speedup Behavior in the Realm of Internode Communication", Proceeding of the IEEE Southeast Conference, Fort Lauderdale. April 2005*

7. *Hennessy, J., Patterson, D., "Computer Architecture, a Quantitative Approach", Third Edition, Morgan Kaufmann, 2002.*

8. *Jain, R., "The Art of Computer System Performance Analysis", John Wiley, 1991*

9. *Koenigs, A. "Industrial Strength Parallel Computing", Morgan Kaufmann, San Francisco, 2000*

10. *Landers, R., "Product Assurance Dictionary", Marlton Publishers, Marlton, NJ, 1997*

11. *Pecht, M., "Product Reliability, Maintainability, and Supportability", CRC Press, NY, 1996*

12. *Schuster, S. "Security with Compromise", Analog Devices, 2005*

13. Vaidyanathan K., Madan, B., Goseva, K., Trivedi, K., "A Method for Modeling and Quantifying the Security Attributes of Intrusion Tolerant Systems", DARPA, 2003