Reliability Engineering - Business Implication, Concepts, and Tools

Dominique A. Heger, Fortuitous Technologies, Austin, TX, (dom@fortuitous.com)

Introduction

An emerging consensus in the systems performance community is that the traditional performance centric focus has become misdirected, and that issues such as reliability, availability, and scalability have emerged as being as important as peak systems performance. The basic term *availability* carries many potential connotations. Traditionally, availability has been defined as a binary metric that basically describes whether a system is up or down. A common extension of this definition is to compute the percentage of time that the system is on average available. *Reliability* on the other hand can be expressed as the probability that a failure situation will not emerge over a certain period of time in the future. The argument made is that reliability and availability are closely related terms, but that they are not interchangeable entities per se, as they represent different expressions revolving around the same issue. While both dimensions are closely related to the overall stability of a system (or an application component), reliability focuses on failure rate based statistics whereas availability represents the measure of time a system or application component is operational. The goal of this primer was to briefly elaborate on the similarities and differences among reliability and availability, and to introduce some of the tools, concepts, and business implications that surround reliability engineering.

Reliability Engineering Concepts

Reliability is the probability of any hardware or software component to function without failure when operated correctly for a given time interval under stated conditions. Hardware and software component failures cost money and cause unreliability issues and concerns. The business focus on reliability revolves around controlling the failure rate to reduce overall costs and to improve operations by enhancing the overall business performance by utilizing affordable levels of reliability.

An interesting artifact of any reliability analysis is the reference to reliability, but actually measuring or quantifying failure rates. A failure situation demonstrates the evidence of a lack of reliability in an infrastructure. Reliability problems are expressed as failures, and failures cost money in any economic enterprise. In most circumstances, failures result into a form of a *downtime scenario*. The argument made is that the process of the *definition of failure* that leads to an increased need for reliability improvements is paramount. Failure situations normally galvanize organizations to take action. The fact of the matter is that in most circumstances, the actual funding for reliability improvements comes from the cost of unreliability. At the center of any reliability improvement study has to be the need to find affordable solutions. Successful reliability engineering work can be described as the perpetual quest for affordable improvements that result in increased profits by solving the most important reliability problems. Effective reliability engineering can not be described as the *quest for perfection*, but rather the search for effective business solutions to resolve the issues that cause the majority of the failure situations.

In most circumstances however, raw reliability numbers do not provide the motivation for actual business improvements. The raw reliability numbers have to be converted into monetary values that express the cost of encountering *unreliability*. Annualizing any losses by means of the cost of unreliability immediately identifies the amount of money that should be available to rectify reliability issues. As with systems performance, purchasing more hardware components without conducting an in-depth analysis of the infrastructure results into working on the symptoms without resolving the actual problem, and has to be considered as being counterproductive in the long run. Reliability requirements fluctuate based on the competitive conditions of the market place. Ergo, reliability values are not immutable as they change along with the business conditions.

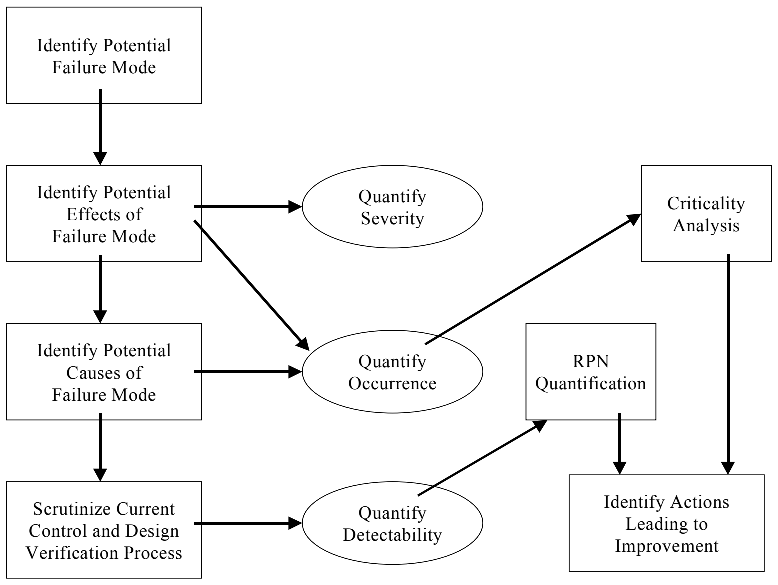
The ramification is that the different business conditions require the use of different reliability engineering tools for resolving the business problems. Reliability engineering tools such as Failure Reporting and Corrective Action Systems (FRACAS), design reviews, decision trees, Failure Mode and

Effects Analysis (FMEA), Fault Tree Analysis (FTA), or Reliability Qualification Tests (RQT) are available to support the effort. The issue is that no company can afford too little or too much reliability, which transforms into another exercise in compromises in regards to the overall systems stability scenario. The cost of unreliability has to be engineered and controlled, which implies that the discussed reliability tools have to be utilized in a cost-effective manner.

Failure Mode and Effects Analysis (FMEA)

The FMEA technique is considered as representing a pervasive reliability engineering methodology that can be applied in a vast variety of areas, focusing on hardware as well as software components, respectively. The intention of FMEA is to identify potential failure modes of any system components, evaluate the impact on the system behavior, and to propose appropriate countermeasures to suppress these effects. To emphasize, the focus has to be on failure prevention and not on failure detection per se. The FMEA technique is well established at a system and hardware component level where the potential failure modes are usually known and can easily be quantified. In other words, FMEA represents a systematic way of identifying any potential failure modes of a system or function, and allows evaluating the effects of the failure modes on any (potential) higher level of abstraction. The objective is to determine the cause for the failure modes and what could be done to eliminate or at least economize on the probability of failure. FMEA is best described as a bottom-up based methodology where the system under scrutiny is hierarchically divided into sub-components. The decomposition has to be conducted in such a way that the failure modes of the components at the bottom level can be identified. The failure effects of the lower level components constitute to the failure modes of the upper level components. After identifying the decomposition approach, the next step is to define the failure mode, the failure effects, as well as the failure cause of the actual component (see Figure 1).

Figure 1: FMEA Roadmap


Identify Potential Failure Mode
Identify Potential Effects of Failure Mode
Quantify Severity
Criticality Analysis
Identify Potential Causes of Failure Mode
Quantify Occurrence
RPN Quantification
Scrutinize Current Control and Design Verification Process
Quantify Detectability
Identify Actions Leading to Improvement

For each component, the severity, the occurrence, and in some cases the detection variable has to be quantified. The severity variable represents a rating that corresponds to the seriousness of an effect of a potential failure mode. The occurrence variable depicts a rating that corresponds to the rate at which a first level cause and its resultant failure mode will occur over a certain life span. The detection variable discusses a rating that corresponds to the likelihood that the detection methods or the current control infrastructure will detect the potential failure mode over a certain time span. All three variables are normally expressed on a scale from 1 to 10. In most cases, hardware manufacturers provide the analyst with detailed information on the failure modes and frequencies of their occurrence. The actual interpretation of an FMEA analysis normally revolves either around the Risk Priority Number (RPN) method or focuses on an actual Criticality Analysis. The RPN identifies the greatest areas of concern by comprising the assessment of the severity, the occurrence, and the detection ratings, respectively. In other words, the RPN represents the product of the three variables.

The term Criticality Analysis (CA) on the other hand describes another quantitative extension to the FMEA technique that is only based on the severity of the failure effects and the frequency of the failure occurrence. The CA analysis is considered as being more proactive than the RPN method, as an RPN based approach assigns an equal weight to the detection variable. The argument made is that before an organization should allocate resources to improve the detection mechanism, all opportunities for reducing the occurrence and minimizing the effects of the failure modes should be explored. The recommendation is that a comprehensive analysis should incorporate a combination of the severity level, the CA, and the RPN to decide when corrective actions have to be taken. The thesis made is that the severity level and the CA analysis should always be consulted while conducting an FMEA study. The following experiment (Table 1) elaborates on the different conclusions that might be drawn if an analysis is solely based on the (popular) RPN method:

Table 1: RPN Analysis

Failure Mode	Severity	Occurrence	Detection	RPN
FM1	4	4	10	160
FM2	5	9	2	90
FM3	9	3	1	27

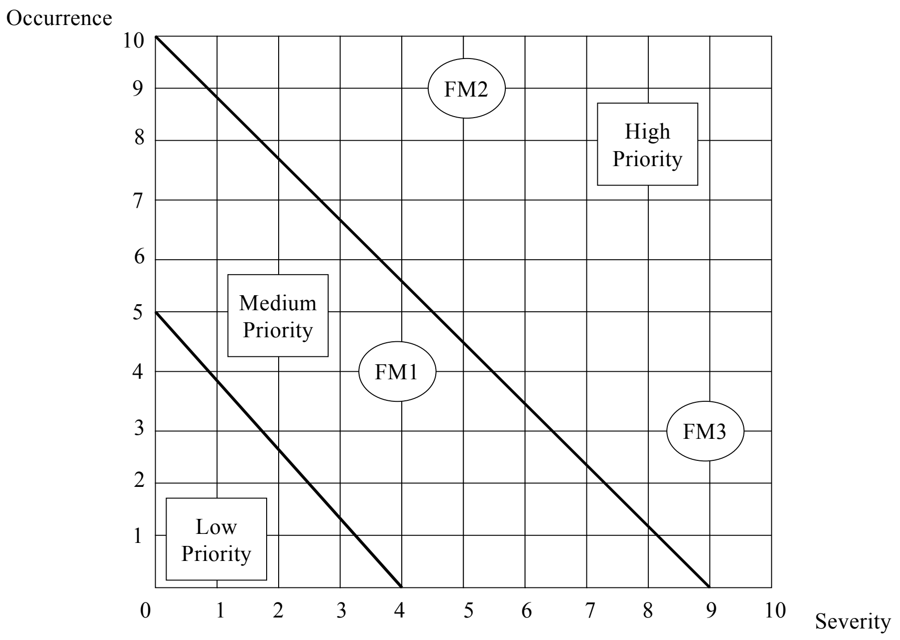
In this particular scenario, FM1 represents the failure mode with the highest RPN. Conducting a CA study focusing on the severity and occurrence variables leads to an area chart that divides the spectrum of the two variables into three sections that represent the high, the medium, and the low priority failure modes (see Figure 2). Based on the area chart, the failure modes FM2 and FM3 represent high priority failure modes, whereas the failure mode with the highest RPN only represents a medium priority failure mode. Quantifying the detection rate is in many circumstances rather ambiguous, ergo the recommendation to always evaluate the RPN in conjunction with a CA study. The severity in itself bears a lot of merit and should always be scrutinized first. A severity of 8, 9, or 10 (10 representing a potentially hazardous failure) should automatically result into launching a *corrective action* process. The strategy for addressing the actual failure modes should follow the following layout:

1. Eliminate the occurrence
2. Reduce the severity
3. Reduce the occurrence
4. Improve the detection
5. Provide a means of detection

On the software side, the failure modes are generally unknown, as a lot of software modules do not fail in the same way as hardware components. In most circumstances, software components deliver incorrect results and depend on the dynamic behavior of the application workload itself. When considering FMEA for a software application (SWFMEA), the utmost purpose of the analysis usually revolves around identifying the software faults that in some circumstances could jeopardize the proper functionality of the system. It has to be emphasized that compared to hardware components, software modules do not wear out in a physical sense, they deteriorate. The deterioration is not a function of time but a side effect of changes (a function of usage) made in the maintenance phase to correct defects, to adjust the application to changing requirements, or to improve performance.

Further, SWFMEA should not be considered as a replacement for traditional software reliability methods. SWFMEA is intended to be a systematic thinking method, or more specifically a means of anticipating issues and improving the validation process. Never less, FMEA can be utilized in all phases of the system life cycle from the requirement specification to the design, implementation, operation, and maintenance phases, respectively. The largest return on investment can be found in the early phases of the design, where FMEA (DFMEA) can be used to isolate problem areas in the systems structure and thus avoid expensive design changes later on in the life cycle.

Figure 2: Area Chart



To support FMEA in a software environment, the recommendation is to utilize additional techniques such as a Fault Tree Analysis (FTA). As a fault tree based approach is considered a static technique that does not reach all the dynamic aspects of a software subsystem, the FTA technique in itself has to be combined with a Petri Net or a Dynamic Flow Graph analysis, respectively. As already elaborated, software component failure modes are normally unknown as if a failure mode would be known it would hopefully be corrected. Therefore, the definition of failure modes is the biggest challenge while applying FMEA to software components. The analysis has to be based on existing knowledge surrounding the software components and requires postulating the relevant failure modes.

Conclusion

This primer outlined the importance of understanding the interrelationship between availability and reliability while conducting an analysis either on a systems component or an application task level. Reliability is paramount in modern computer systems, in some circumstances even at the expense of systems performance. A slower but reliable system might be acceptable whereas any failure situation or data loss issues might have a catastrophic impact on the user community. Any downtime could potentially be equally unacceptable, which leads to the importance of high systems availability. Further, this report outlined that *availability* and *reliability* are not interchangeable terms, as a system might reveal a high availability but at the same time discloses a low reliability behavior (or vice versa). As an example, a network router is a physical device that does not store any state data. Hence it can be considered as one of the few physical resources where a data loss is acceptable as long as high availability standards are maintained. Every reliability and availability analysis is *environment specific* and has to be conducted by taking the entire computing infrastructure into consideration. Analyzing and scrutinize the impact that any potential changes will have on systems performance and systems maintenance is paramount. Dependability, scalability, as well as maintainability components impact the overall systems stability. Only a high ranking in all three dimensions will result into an overall useful, trustworthy, and effective computing environment.

References

1. Dellin, T., “*Yield and Reliability*”, Sandia National Laboratories, 1998

2. Dugan, J., "*On Measurement and Modeling of Computer Systems Dependability*", IEEE Transactions on Reliability, Vol. 39, Oct. 1990.

3. Edmiston, P., Peng, J., “*Reliability and Availability of Networked Client-Server Systems*, CMG, 1993

4. Johnson, M., Malek, M., "*Survey of Software Tools for Evaluating Reliability, Availability, and Serviceability*," ACM Computing Vol. 20, Dec. 1988.

5. Palady, P., “*FMEA – Author’s Edition*”, 2nd Edition, 1997

6. Pentti, H., Atte, H., “*Failure Mode and Effects Analysis of Software Based Automation Systems*”, VTT, 2002

7. Price., K., “*Failure Mode & Effects Analysis in Software Development*”, SAE Technical Paper, 1998

8. Sahner R., Trivedi, K., "*Reliability Modeling*" IEEE Transactions on Reliability, June 1987.

9. BU, “*Effective Processing Time Modeling*”, Boston University, 2002