White Paper

Mahindra Satyam

Software FMEA and Software FTA – An Effective Tool for Embedded Software Quality Assurance

About the Author

Dr. T Chitra Rajan
Safety and Reliability Expert - Embedded Practice
Integrated Engineering Solutions

The author leads the Safety and Reliability team in embedded systems for various industry domains including automotive, aerospace, Medical and E&U. With over 25 years of technical experience with Defense Research organizations, the author prime area of focus has been in the analysis of Safety and Reliability of indigenously designed and developed defense systems for the armed forces. The author has also evaluated simulated weapon systems and analyzed statistical data for over 10 years.

With over 35 publications in National/ International Conferences/ Seminars/ Journals, the author holds a doctoral degree from Indian Institute of Science Bangalore, India and Masters degree from University of Georgia, USA. She has also co-authored a book on Reliability and Six Sigma published by Springer Verlag in Feb 2006.

Abstract

One of the most important activities in the software development process is the software quality assurance. The software quality assurance consists of activities such as design walk throughs, testing and inspections. These activities are carried out in the following phases namely, functional requirement specifications, software design, detailed design and coding. Software FMEA (Failure Mode Effect Analysis) and Software FTA (Fault Tree Analysis) serves as an effective tool in these activities in reducing component level testing effort and also plan an effective integration and system testing. The functional level software FMEA carried out during the detailed design phase helps in identifying the structural weakness in the software design. Once the coding phase is over software FTA and variable level software FMEA can be carried out to study the effect of failures. Software FTA helps in identifying the critical variables responsible for an undesired event. Software FMEA carried out at variable level helps analyze the code in detail for the various failure modes of the variables leading to critical events which are of concern from the functional requirement of the system. These activities help in modifying the code at an early stage so that the costly modifications at a later stage could be avoided.

This paper discusses the details of software FMEA and software FTA which are effective in the software quality assurance phase with an example.

Introduction

The electronic industry has transitioned from hardware stage to the embedded software stage to increase the flexibility within the product. Introduction of embedded systems has helped to meet the customer requirements with a few additional lines of code in the software. This in turn has helped in getting over the obsolescence and keeping pace with the customer demands in the market. Right from consumer electronics we use in our daily life to any high end system (e.g., air and railway traffic control, nuclear plant control, aircraft and car control), embedded software plays the key role in all the applications. If we consider the automotive industry 90% of innovations are driven by embedded systems. The vehicle costs are function of the electronics and software used in the car and nearly 50 to 70% of

Inside the issue

the development costs for ECUs are related to software. The increasing complexity of software in all these applications calls for a high level quality assurance before the product is rolled out. While we impose a tight quality assurance activity one has to take care of the time and cost involved in it to meet the time to market goal as well.

The challenge with the embedded software is that it executes in real time. Safety critical and performance critical applications require exact limits and measurements. Statistics or averaging measurement methods are not suitable. Reviews, inspections and testing are accepted as quality improvement techniques in the software at every stage in software life cycle. Overall the safety critical and performance critical nature of many embedded control application is the paramount reason for demanding a performance software quality tools. Antilock Braking System, instrument landing system, telecommunication network for spacecrafts, patient monitoring systems and a host of other products on which we are now so heavily dependent on embedded software must prove that they deserve our confidence in their safety and reliability. In this context an embedded system is a little program that carries an awful lot of responsibilities.

Safety critical and performance critical applications require exact limits and measurements.

This paper describes the application of Software FTA (Fault Tree Analysis) and Software FMEA (Failure Mode Effect Analysis) as one of the quality assurance tool. These analyses are discussed from the experience gained by application of these techniques extensively in the automotive industry. Software testing generally aids in identifying the faults where as the Software FTA and FMEA addresses the failures.. These analyses aid in identifying potential failures at the system level that is to say the failures with respect to the system functionality for which the system is designed for. The FTA and FMEA analysis aids in identifying the single point failure causes and the combination of conditions leading the undesired failure events. Once we understand the conditions leading to the functional failure, decisions can be made to mitigate the risks associated with the failures. Eliminating the risks or mitigating the risks is a very important task for a safety critical and performance critical applications.

Software FTA and FMEA

There are two techniques used by hardware which are adopted for software. They are

- Fault Tree Analysis (FTA)
- Failure Mode Effects Analysis (FMEA)

While these analyses are more predominant with hazard analysis, they are excellent tools for determining the weak areas of the software system. These analyses help identifying the potential failures that may lead to potential system failures. The results of the analyses helps in protecting against those potential failure modes

The FTA is a top down approach and FMEA is a bottom up approach. Both these analyses provide input in testing, verification and validation. Both these methods addresses the effect of failures at the system functionality level . The out come of the analysis provides inputs for component testing, integration tests and system testing. Software failures happen due to memory corruption, incomplete execution, timing errors (early execution / late execution) etc. The FTA and FMEA analysis aids in identifying the effects of these failures and look for the protection provided in the code for some of the critical failures. The functions/ variables for which such protection is not provided are identified before the product is rolled out.

Software FTA

In order to carry out the software FTA we identify the undesired events with respect to the system functionality for which the system is designed for. If it is a safety critical system these undesired events refer to hazards which are foreseen with respect to the system functionality and also the environment and other operating conditions in which the system is operating. Taking the undesired event as the top most events we logically connect the software components leading to the event. Starting from the software functions leading to the event we come down logically to the conditions triggering these functions and then the global variables responsible for the triggering. The stopping criteria for the analysis in general are the global variables leading to the event. The cut set analysis provides the single point variables and the combination of several other variables leading to the event.

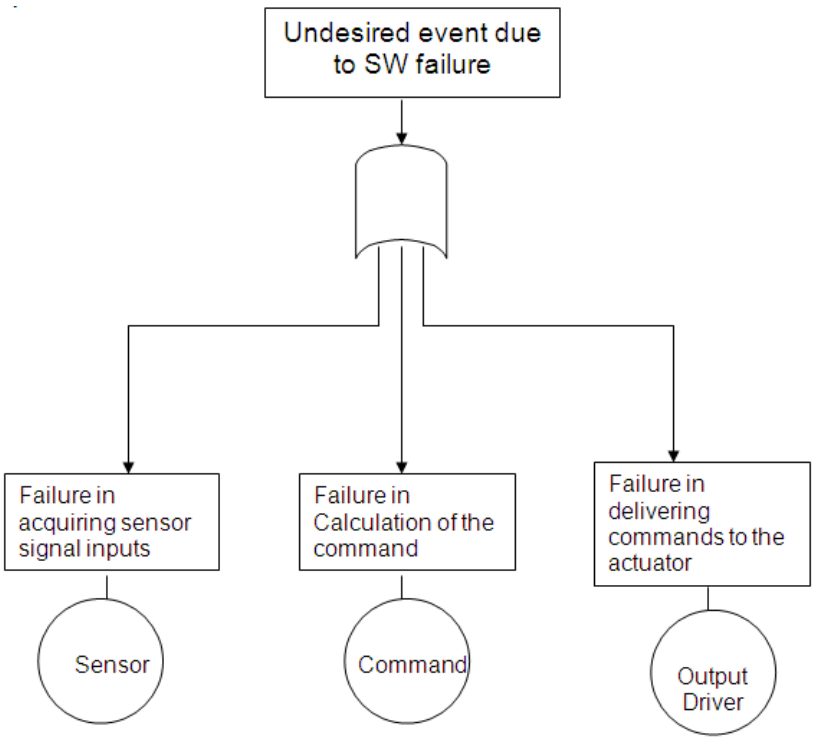
Taking the undesired event as the top most events we logically connect the software components leading to the event.

For each of the single point variable we look for the protection provided in the code. For example the value may be copied in another location or a periodic RAM check done in order to identify the memory corruption etc. Where the cause is due to failure of group of variables, the common cause is identified to look for the simultaneous failure of the group of variables at the same instance. Thus when we carry out the FTA we go through the entire code and one could give a judgment analysis as to whether the critical variables are protected. Depending on the role of the variable in the code additional tests can be suggested.

Thus Software FTA is used as one of the verification tool. Software FTA might be applied to software design representations to locate problems early and the same can be fixed before the design is frozen.

Example:

The following example gives the general layout in carrying out the software FTA for an embedded system.



Software FMEA

Software FMEA is a detailed analysis of software components (functions and variables). FMEA analyses component interaction, dependencies between data related failures. Software FMEA is performed in two levels namely functional level and variable level

Functional level FMEA is used in identifying structural weakness in the software design. It also helps reveal weak or missing requirements and latent software. The functions of the same are listed below. Here it is important to decide upto what level each higher level functions are broken further into their constituent functions in order to select the candidates for FMEA. There is no fixed rule on this however normally we go down up to three or four levels. One needs to make a decision on the importance of those functions. Normally the failure modes considered are function fails to execute, incomplete execution, timing error (too early, too late), erroneous execution. For interrupt service routines the failures considered are failure to return thus blocking lower priority interrupts from executing and returns incorrect priority. The effects of these failures are analysed for their system effects. Based on the criticality of these failure effects the software design is examined for the protections mechanisms provided.

Based on the criticality of these failure effects the software design is examined for the protections mechanisms provided.

Example:

Consider an example of functional FMEA of the function written for speedometer control.. The function analysed is 'read vehicle speed'. The failure modes considered are fails to execute and erroneous execution of the function. The system effect due to the failure leads to a severity of 10. (Severity numbers are assigned 1 to 10. 10 refers to the highest order of severity and one refers to least). With the implantation of the monitor checks the severity is expected to reduce to 7.

Software Element	Failure Mode	Local Effect	System Effect	Pot. Severity	Recommendation	Projected severity
Read VehSpeed()	Fails to Execute	No CAN Vehicle speed signals are read	System will continue to use the last read Vehicle speed value and filtered output calculated based on that value. Since the last read vehicle speed values are within range, DIAGNOSTIC function will not detect the fault. This will update the wrong value in Electrical angle output and will incorrectly move the pointer in Speedometer and will indicate wrong Vehicle speed than the actual speed.	10	A software execution monitor that checks the execution of this software element needs to be employed	7
Read VehSpeed()	Erroneous Execution	Vehicle speed signal values are incorrectly read	SpeedSignalValid() function will catch any out--of-range signal values. However if the CAN signal values are within range, system will continue to use the erroneous Vehicle speed signal value and hence output will be incorrect. Potentially incorrect vehicle speed will be indicated in the Speedometer misguiding the vehicle operator.	10	This could be caused due to any memory byte corruption. Need to have checks that monitor memory cells.	7

Detailed level FMEA

Is carried out when the complete code is available. Based on the variable type the associated failure modes are analysed. The values of the analog variables may take a higher or lower value than the prescribed value at a given instant of time due to bit flips. The Boolean variable may take the value true when it has to be false and vice versa. The enumerated variables has a set of values to take and the failure mode considered will be the variable taking a different value than the one it has to have at a particular instant of time. In addition to potential variable failure modes, potential software processing logic failure modes may be considered. This involves examining the operators such as addition, subtraction, comparison etc to determine the possible negative effects that must be addressed. By assigning severity (severity values are assigned from 1 to 10 indicating 10 is most severe and 1 least severe) to each failure mode with respect to system effect we can identify the critical variables. Then these critical variables are analysed for the existing protection provided in the code. If the protection not provided then the same is suggested for implementation. The analysis provides critical variable list, unused variable lists along with the recommendations to avoid failures leading to failure of the system functionalities.

The analysis provides critical variable list, unused variable lists along with the recommendations to avoid failures leading to failure of the system functionalities.

Example :

Consider a variable level FMEA carried out on a variable calculating the vehicle speed value. The system effect leads to a severity level of 10. With the implementation of the fault monitor periodic memory check it is expected that the severity will be reduced to 7.

Variables	Failure Mode	Purpose	Local Effect	System Effect	Pot. Severity	Recommendation	Project Ed severity
Last_spd_filtered_data	Memory Corruption	Vehicle Speed filtered Output. This output is used to physically drive the speedometer pointer to the correct Position.	Corruption of this variable will update wrong value in Electrical angle output variable, which will be used to physically move the Speedometer pointer and will indicate wrong Vehicle speed.	In worst case, If the Vehicle Speed Output value erroneously corrupted to lower value (When there is increase in speed), Speedometer will inform the driver of low Vehicle speed than the actual speed, which misguides the driver.	10	Fault Monitor Periodic Memory check	7

Conclusion

The combined results of software FMEA and software FTA provide input for analysis of interdependencies (causal/temporal) justification for prioritization of verification/ validation test systematic approach from system down to SW subsystems. Since the software FMEA and FTA analyses require thorough understanding of the software design, system functionality and the code, certain efforts on component level testing can be reduced. Some of the findings can aid in better planning of integration and system testing.

For further information please write to **rfi@mahindrasatyam.com**