An Experimental Safety Analysis using SFMEA for a Small Embedded Computer Control System

K.Jayasri
Department of Computer Science and Engineering
GMR Institute of Technology, Rajam, Andhra Prasad, India

P. Seetharamaiah
Department of Computer Science and Systems Engineering
Andhra University, Visakhapatnam, Andhra Prasad, India

Abstract—Software Failure Modes and Effects Analysis (SFMEA) is a system safety analysis technique which is widely used in the aerospace, automotive and other safety-critical intensive systems. FMEA methods are difficult to identify and analyze the failure modes which are caused by the dynamic logical information between interfaces or functions, such as software-hardware interaction. To intuitively assume the effects of module failures in a system, numerous approaches have been proposed. This work addresses the use of SFMEA by performing experiment for safety-critical embedded control systems. The work presented here provides a generic example illustrating the application of SFMEA to a microprocessor based computer control system having little or no hardware protection. This paper demonstrates the application of Functional FMEA, interface FMEA, and detailed software FMEAs into safety-critical software system. Through the SFMEA method, the hardware failures and software faults are addressed. The safety analysis reveals several design deficiencies and physical faults for which modifications are proposed. This paper also reveals that, when properly implemented SFMEA at the right point in the Software Development Life cycle, it makes requirements, design and code reviews more effective. It also identifies single point failures due to software. The work presented in this paper can be generalized and applied for future use of designers in the field of any safety-critical embedded computer control systems.

Keywords: FMEA, safety analysis, safety-critical system, SFMEA

I. INTRODUCTION

Software plays the essential and focal role in the functioningof different sorts of systems like real-time computer control systems, embedded computer systems, etc.In fact, softwareis progressively being utilized to handle Safety-Critical Computer Systems (SCCS)which were once in the past controlled by people or hardware. With the expanded reliance of systems on software and with its developing size and unpredictability, it has now turned into the major contributor to system failures and a primary threat tosystem safety and reliability.

The FMEA methodology is one of the risk analysis techniques recommended by global norms. It is a methodical procedure to recognize possible failures existed in an embedded computer system. FMEA is mostly adapted for equipment and material failures, however in a wide sense, human blunder, performance and software errors can likewise be incorporated. By applying the SFMEA procedure amid the different phases of a product's lifecycle, the approach gives an efficient system to looking at all the routes in which a product can fail. The results of SFMEA in turn affect the product design, process development.

Nathaniel Ozarin (2004), FMEA strategy first recognizes the failure modes and after that analyses the cause and effect of these failure modes. K. Jenab and J. Pineau (2015). In the practice of a SFMEA, analysts gather lists of module failure modes and try to deduce the effects of those failure modes on the system. Although there is software available that assists engineers in performing clerical tasks, such as forming tables and filling data. The intelligent part of an FMEA process remains a manual and difficult process. Thus, one of the main criticisms of SFMEA is that the time taken to perform the analysis can often exceed the period of the design and development phases and therefore the analysis defect becomes a mere deliverable to the customer and not a useful tool capable of improving the design. SFMEA is an import and commonly used method to improve the safety and reliability of the safety-critical software. Experimental platforms are key elements for the practical approach of hypothesis and theories in control systems. One of the main contributions of this experiment is the fact that they are a main tool for the technological transfer and the innovation process. The process of SFMEA has three main focuses:

- The acknowledgment and assessment of potential failures and their effects.

- The identification and prioritization of activities that could eliminate the potential failures reduce their chances of occurring or reduce their risks.
- The documentation of these identification, assessment and corrective exercises so that product quality improves over time.

Isaksen(1997) depicted in a technical report as, SFMEA is a bottom-up safety analysis of potential failure modes inside a computer system and assessment of the related impacts on system usefulness. It is utilized to distinguish potential design shortcomings such that they can be moderated in the early phases of a design program. Shortcomings inside the software product segment of systems are of specific concern. However as software increments in size and intricacy it cannot typically be exhaustively verifiedas a result, latent software faults have become a leading source of safety and quality issues for medical devices. (OSEL annual report 2011), over 20% of all medical device recalls in the United States were due to software failures. R.T. Anderson,(1976), MIL-STD-1543 (1974) and SAMSO-STD 77-2, (1977) Standards and a handbook has been published to direct its orderly application on military projects. The standards give useful insight into why FMEA is employed.

According to Goble.W (2012), FMEA was made in the 1960s as a major aspect of the U.S. Minuteman rocket program so as to discover and relieve unforeseen design issues. McKinney proposed that with the goal FMECA should be viable, it must be executed right on time being developed so design might be modified to moderate or dispense with the calamitous, critical, and safety related failurespossibilities. Jenab, K., (2005) built up a group based FMEA which attempted to determine struggle among experts. J. B. Bowles (2001), Chao and Ishii (2007) introduced an illustration and contextual investigation utilizing an adjusted FMEA design process. It decomposed the design process into six potential problem areas and used a question-based FMEA approach.

Dale (2009) composed a book which characterizes SCCSs and the procedures which can be utilized to solveissues identified with safety. Bozzano (2010) added to the safety assessment of critical systems with a book concentrated on procedures and strategies for reliability, dependability, and safety assessment. Illiashenko (2012) declared that there is no universal substantial methodology for figuring out which procedure to use for reliability analysis. Their study had two primary objectives; decrease the risk of incorrect safety evaluation, and inspect FMECA-based methods to decide how and when to utilize them for specific tasks. De Miguel (2005) and Franceschini.F (2001) and Haider, A. (2013) expressed that utilization of only one analysis procedure is deficient, and suggest combined usage of methods is important for safety analysis of critical systems.

Early researches described limited experimental concepts.They depicted more hypothetical ideas. This paper depicts an experimental safety analysis using SFMEA for Safety-Critical Computer Systems. At the point when appropriately executed SFMEA at the right point in the Software Development Life cycle, it makes necessities, design and code reviews more viable. It likewise recognizes single point failures because of software.

The rest of the paper is organized as follows. The experimental setup and implementation is described in section 2. SFMEA with results described in Section 3 and section 4 gives conclusion.

II. EMBEDDED COMPUTER BASED BALL POSITION CONTROL SYSTEM (BPCS)

The goal of this research work is to edify the utilization of SFMEA for an embedded control system through the advancement of an experiment. P. L. Goddard (1993), (2000) talked about the methodology for SFMEA together with perceiving the sorts of variables and their failure modes. Experimentation of the proposed procedure depends on the ball position controller system, which is appeared in figure 1. The control objective of this work is to direct the stream of air into a plastic tube to keep a little light weight ball suspended at a predetermined height called the set-point. Expanding the flow raises the ball and diminishing the flow brings down it. The BPCS test comprises of: 2-feet long white plastic tube, light weight ball, Dc engine fan, and infrared sensor circuit and 89S52 micro controller.

The vertical 2-feet long clear plastic tube joined to a stand, which contains a light weight ball inside, a Dc engine fan at the base to lift the ball, and an infrared sensor at the top to sense the ball's height. The tube is associated with the Dc engine fan deltas by means of an input manifold which has a channel at the base as demonstrated. There is an output manifold at the highest point of the plastic tube with an outlet as appeared. The presence of the manifolds is a key part of the experiment. The infrared sensor hardware identifies the position of the light weight ball and the microcontroller directs the power supply connected to the Dc engine fan in order to control the air flow into the white plastic tube, keeping the light weight ball at a foreordained height.

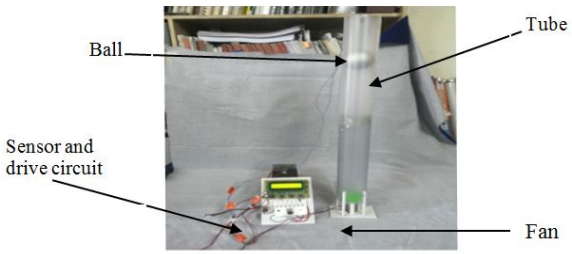

Figure 1. Ball Position Control System (BPCS)

BPCS Explanation

The light weight ball position control system experiment is a system made out of five modules, where one of them has a Dc engine fan to blow air into the white plastic tube moving a Styrofoam light weight ball inside it as appeared in figure 1.The modules are instrumented with an infrared sensor to sense the light weight ball height. The air flux originating from the Dc engine fan is exchanged to the white plastic tube utilizing a pressurization cylinder whose goal is to diminish the turbulence created by the dc engine fan, which conveys a practically laminar flux. Every module is combined with the others by the operational target of the system is to keep up the light weight ball at a predefined height in the plastic tube. The software in the microprocessor gives the control mechanism; it executes a corresponding controller that is, the output signal data is corresponding to the measure of error in the ball's position with respect to the set-point. The software is actualized in low level programming, which is collected and downloaded into the system inside ram of microcontroller. This guarantees the project stays in the microcontroller when the system power is out.

A Block diagram of the BPCS system is appeared in figure 2. Every module is combined with the others by a typical complex. The base box compares to the input manifold. The airflows into the manifold by the unique input located at the left side of the box. The air in the input manifold is circulated over every module in a parallel way. Depending on the force connected by the dc engine fan and the input size of the manifold, the air flux proceeds with its direction moving the ball inside it. The air from the plastic tube is consolidated again in the output manifold and shot out through the output, in the right half of the case. This reconfigurable structure possesses input and output manifolds in individual boxes that can be associated between them by their design as a Lego piece. The BPCS dynamical model contains a vitality exchange via airflow from the dc engine fan to the light weight ball. This exchange is regularly nonlinear.

The microprocessor creates a Pulse Width Modulation (PWM), it is a modulation technique used to encode a message into a pulsing signal. Despite the fact that this balance strategy can be utilized to encode data for transmission, its primary use is to permit the control of the power supplied to electrical gadgets, particularly to inertial loads, for example, motors. PWM creates a control sign to direct the rate of the dc motor fan. The higher the duty cycle of the PWM signal, the speedier the fan runs. An output control circuit between the chip and the Dc engine fan gives equipment "stops" that guarantees that the PWM sign is inside the predetermined "safe" working extent.

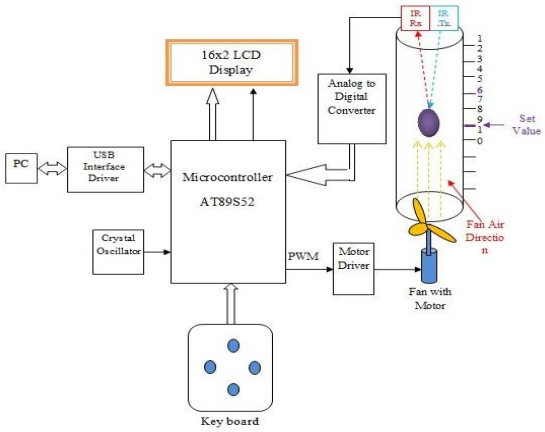

Figure 2. Block diagram of Ball Position Control System (BPCS)

On the off chance that the Pulse width Modulated cycle is out of the meticulous range, the flaw pointer light is exchanged on and the system is shut. This fault confirmation capacity guarantees a safe functional range for the DC-engine even in the event of a software malfunction. The elevation of the ball is purposeful by the sensor. In perspective of the way that the ball is white, the infrared beam from an emitter is started from its façade into the sensor. On the off chance that the ball is closer to the sensor, light is gotten with high quality and the output sign is likewise morestronger. The outputpower of the sensor is sample by the converter Analog to Digital, which is implanted in the microprocessor. The internal power is changed into a one-byte value in an ADC register, which exhibits height of the ball. The ADC is standardized so that a 016 corresponds to the ball being at the base of the tube, and FF16 corresponds to the ball at apex. The error of the ball point is calculated by deducting the thought ADC value from the set-point esteem. The variation is increased by a constant to decide the correction to the PWM signal. If the intensity of the error is high, then more rectification is required. If the calculated ADC value is below the set-point variable, the ball is under the set-point and the microprocessor enlarges the duty cycle to raise the fan and ball speed. On the other hand, if it is under the set-point, the duty cycle is condensed to lower the ball. The input control circuit sandwiched between the infrared sensor and the microprocessor make sure that the voltage at the input port is not larger than 5 volts. The I/O control circuits decrease the risk of damage to the system from unforeseen infrared sensor failures. For example, if the sensor shorts out and outputs an enormously high power (e.g.10V) to the microprocessor without a safety control circuit, then the microprocessor would be harshly smashed and have to be changed.

Figure 3 shows the program flow chart for the ball position control system. In the flow chart, first initialization of LCD, initialization of serial port and initialization of PWM variable are done. Then the software enters an infinite loop in which the analogy voltage from the sensor is read, converted to a number, and stored in an ADC register. The duty cycle is then calculated and the output pin set, high or low, to achieve this cycle. The timer interrupt in the microprocessor triggers an ISR every 5microseconds: a counter, which is initially set to 100, is decremented by 1 on each interrupt. To achieve a given duty cycle (say 50%) for the PWM signal, the output pin is set low for each interrupt until the counter reaches the given value(i.e., 50) then it is set to high until the counter is decremented to 0. At 0 the counter is reset to 100 and the output pin to low and the cycle is repeated, thus producing a 0.5 Ms (mille second) period with a 50% duty cycle.

III. SFMEA FOR BPCS

During the time spent a SFMEA, analysts incorporate lists of modules failure modes and try to gather the impacts of those failure modes on the system. System models, commonly simple engineering diagrams, assist analysts in seeing how the local impacts of modules failures proliferate through complex structures and at last cause risky impacts at system level. The system has five principle functional modules: 1) Ball-position-controller 2) Dc engine fan and drive circuit 3) Infrared sensor 4) Display module 5) USB-Interface.

In table 1, if the output signal pin stays high, the duty cycle goes to 100% and the fan blows at the full speed. This results in ball shooting to the top of the plastic tube, possibly damaging the sensor. If the output pin stays low, the duty cycle goes to 0, and the dc motor fan speed decreases and stops. This results in ball falling to the bottom of the plastic tube. If there is no output signal, the effect is the same as if the signal is low and the system loses response, stops, and the ball falls to the bottom of the plastic tube. Observe that the last two failure modes in table 1 have the same level and system effects and thus are put into the same fault category class. The first failure mode is different and hence goes into a different fault category class.

Functional FMEA for BPCS: A functional FMEA analyses the overall system failures. Figure 3 shows the flow chart for the main program of the Ball Position Control System (BPCS).

Functional SFMEA for BPCS: The functional SFMEA shows significant software faults as failures of the functions executed by the software. From the functional point of view the program is spitted into three software tasks, which are consider independently. These are 1) A/D conversion. 2) Output response computation. 3) Interrupt service routine. The Interrupt Service routine flow chart is shown in figure 4. The functional SFMEA views failure modes, component effect, related failure impact and fault category formed by the software. Table 1 shows the functional SFMEA for BPCS.

Table-1 Functional SFMEAFor Ball Position Controller Module

Module	Failure mode	Component Effect	Failure impact	Fault Category
Ball Position Controller	Output signal too high	Dc motor fan runs too fast	Light weight ball is shot to the top of the plastic tube and possibly damages sensor	A
	Output signal too low	Dc motor fan runs too slow	Light weight ball falls to the bottom of the plastic tube	B
	Loss of output signal to drive circuit	Dc motor fan does not run or system does not respond	Light weight ball falls to the bottom of the plastic tube	B

Table-2Analogto Digital Conversion Functional SFMEA

Module	Failure mode	componen t Effect	Failure impact	Fault Category	Observation s
Analog to Digital converter	Failure to get or convert position (in hardwa re)	Incorrect output data	Dc motor fan speed does not change	D	Setup hardware or software validation to ensure the input data is updated periodically
	Too much delay in getting value	Response data too slow	Dc motor fan speed is incorre ct	D	Set up software validation to measure the pin transition time.
	Incorrec t value in interrup t register	Incorrect output data	Dc motor fan speed is incorre ct	C	Set up software validation to ensure the value within the tolerance

The safety analysis of the code shows in table 2. For example, in the first row of table 2, when the input pin is not initialized the effect is that the pin cannot be read. The scribbled pin results in no output data to the Dc motor fan, so the fan does not run, which leads to the consequence that the ball stays or falls to the bottom of the plastic tube. Observe that all three functional failures in Table 2 have the same level system effects and thus belong to the same fault category. The observation section recommends installing is redundant sensor to detect the ball's location and restart the system.

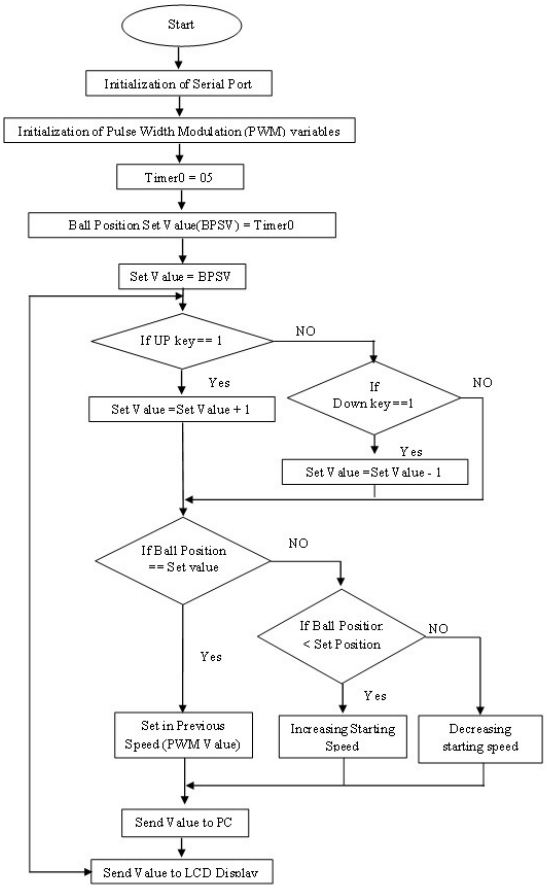

Figure 3. Flow chart for the main program of the Ball Position Control System (BPCS)

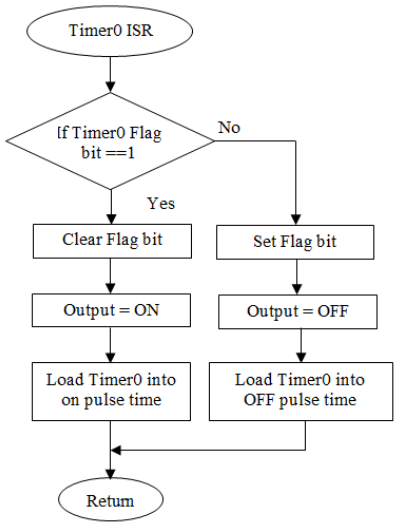

Figure 4. Flow chart for the ISR of the Ball Position Control System

III.1. Analog to Digital Conversion Functional FMEA:

Another functional failure is the conversion of the analog input to digital equivalent (A/D). This has three possible failure modes as shown in Table 2. These are Failure to get or convert position (in hardware), too much delay in getting value and incorrect value in interrupt register. For the first failure mode, it is found that,

the ADC fails to convert the ball position, the result is an erroneous microprocessor output to the Dc motor fan, which itself, leads to an uncontrolled fan, which leads to an out of control system and failure. SFMEA may include some hardware failures as part of the analysis. It is also observed that the last two failure modes have the similar effects the first is different and these effects also differ from those in previous table. Therefore these faults are put into different fault category classes (i.e., A, B, C and D). The second module of the BPCS is an analog to digital converter. Table 2 display failure modes, component effect, and failure impact on the system and fault category.

III.2. Interface Software FMEA

An Interface SFMEA analyses failures affecting the interfaces between software modules. For the ball-position-controller the analysis concentrates on failures of the interfaces between A/D conversion and the error detection module, Interrupt Service Routine and the generation of the Pulse width modulation signal, with Program and OS.

Table-3 Interface FMEAFor The Interfaces Between Interrupt Service Subroutine And Pwm Generation

Interface	Failure mode	Component Effect	Failure Impact	Fault Category	Explanation
Interrupt service subroutine / PWM Generation (Digitize output)	Input duty Cycle to ISR stuck high	Dc motor fan runs too fast	Light weight ball is shot to the top of the plastic tube and possibly damage sensor	A	Set up hardware or software check to ensure the light weight ball does not rise more than some tolerance value above the set-point
	Input duty Cycle to ISR stuck low	Dc motor fan does not run or runs too slow	Light weight ball falls to bottom of plastic tube	B	Set up hardware or software validation to ensure the input voltage within the tolerance
	Pin P0.1stuck high	Dc motor fan runs too fast	Light weight ball is shot to the top of the plastic tube and possibly damages sensor	A	Setup hardware or software check to ensure the ball does not rise more than some tolerance value above the set-point
	Pin P0.1 stuck low	Dc motor fan does not run or run too slow	Light weight ball falls to bottom of plastic tube	B	Set up hardware or software validation to ensure the output voltage within the tolerance

Here we observe the possible failures of the interface between the ISR and the generation of the PWM signal shown in Table 3. Consider the main failure mode Data register holds the proposed duty cycle for the PWM signal; if its value is wrongly high, the Dc engine fan blows at too high a speed. This results in the light weight ball shooting to the highest point of the plastic tube. One restorative action to make because of this failure is to set up a hardware or software check to guarantee that the ball's rise is within a specific heighttolerance of the set-point. In the event that it rises more than the resistance value over the set-point, an error shouldbe motioned to the operator. Watch that this failure mode has the same impacts as failure mode in Table 1; therefore it is allocated to the same fault category.

III.3. Detailed Software FMEA

Detailed SFMEA examines the consequence of individual software variable failures on the system output. The SFMEA shows the effects of different variable errors. This work concentrates only the extreme case (high or low) for each variable. The ball position controller program contains the following software components. These are Interrupt Service Routine Initialization, Analog to Digital conversion subroutine and output response

computation.The experiment founds the various remarks relate to the safety analysis, which shows the effects of the variable failures on the critical software variables in the program If assuming a value of 30 in the set-point variable generates a 30% duty cycle, if the set-point value exceeds the allowed tolerance high (> 70% duty cycle), the ball will stabilize at some point above the set-point. If the set-point value exceeds the allowed tolerance low (<20 % duty cycle), the ball will stabilize at some point below the set-point. The error of the ball point is calculated by deducting the deliberated ADC value from the set-point value. If the set-point is high the value of the error will always be low. The other critical software variables will not be affected by the set-point variable being erroneously high causing the fan speed to be increased. Similarly, if the set-point is low the value of the error will always be high causing the fan speed to be decreased. The other software variables not affected by the set-point variable being erroneously high.

Table 4 Detailed SFMEAfor Output Response Computation

		Critical Software Variable					
Software variables	**Failure Mode**	**Set Height(30 % duty cycle)**	**Error (Frequentl y within 45)**	**Constant for p-controller (Constant =1/3)**	**Previous duty cycle (Around 40)**	**Present duty cycle after updating (Around 50)**	**Constant for converting the scale from 0-100 to $00-$FF (Constant=100/256)**
set-point	**Value exceeds allowed tolerance high**	**>= 70**	**45 low**	**1/3**	**40**	**50**	**100/256**
	Value exceeds allowed tolerance low	**<=20**	**50 high**	**1/3**	**40**	**50**	**100/256**
Error	**Value exceeds allowed tolerance high**	**30**	**$7FFF (16 bits) or a big positive. num.**	**1/3**	**40**	**50**	**100/256**
	Value exceeds allowed tolerance low	**30**	**$8000(16 bits) or a big negative. Num.**	**1/3**	**40**	**50**	**100/256**
Constant for p-Controller	**Value exceeds allowed tolerance high**	**30**	**50**	**$7FFF (16 bits) or a big positive. num.**	**40**	**50**	**100/256**
	Value exceeds allowed tolerance low	**30**	**50**	**$8000(16 bits) or a big negative. Num.**	**40**	**50**	**100/256**
Previous duty cycle	**Value exceeds allowed tolerance high**	**30**	**50**	**1/3**	**>=70**	**50**	**100/256**
	Value exceeds allowed tolerance low	**30**	**50**	**1/3**	**<=20**	**50**	**100/256**
Present duty cycle after updating	**Value exceeds allowed tolerance high**	**30**	**50**	**1/3**	**40**	**50**	**$7FFF (16 bits) or a big positive num.**
	Value exceeds allowed tolerance low	**30**	**50**	**1/3**	**40**	**50**	**$8000(16 bits) or a big negative num.**

Constant for converting the scale	**Value exceeds allowed tolerance high**	**30**	**50**	**1/3**	**40**	**50**	**100/256**
	Value exceeds allowed tolerance low	**30**	**50**	**1/3**	**40**	**>=20**	**100/256**

IV. CONCLUSION

FMEA is a bottom-up analysis of potential failure modes within a system and assessment of the associated effects on system functionality. It is used to identify potential design weaknesses such that they can be mitigated in the early stages of a design program. This paper describes the importance of SFMEA by conducting an experiment on a microprocessor based control system having tiny or no hardware safeguard. The objective of this work is to regulate the flow of air into a plastic tube so as to keep a tiny less weight ball suspended at a predetermined height called the set-point. This paper also demonstrates the functional, interface, and detailed software FMEAs. In SFMEA the software is split into various functional blocks and their failure modes are presumed and the consequences of these failures on the system are determined. The interface SFMEA illustrates faults affecting the interfaces.

The detailed SFMEA examines software faults associated to the key software variables in the program. The experiment founds the various remarks relate to the safety analysis, which shows the effects of the variable failures on the critical software variables in the program. If assuming a value of 30 in the set-point variable generates a 30% duty cycle, if the set-point value exceeds the allowed tolerance high (> 70% duty cycle), the ball will stabilize at some point above the set-point. The error is calculated by subtracting the ball's position from the set-point. If the set-point is high the value of the error will always be low. SFMEA Identified potential failure modes in BPCS, and also assess the risk associated with those failure modes and prioritize issues for corrective action and carry out corrective actions to address the most serious concerns. It is also found that many of the errors found in the software failure modes analysis are likely hardware errors and those may not be detected in the normal operation of the system unless extra monitoring. This work can be extended by reinventing the experiment so that new physical characteristics are included. In this way, SFMEA can be applicable to real time embedded computer control systems and finds its future scope in the applications related to safety critical embedded computer control systems.

REFERENCE

[1] Isaksen, U., Bowen, J.P. and Nissanke, N. (1997) "System and Software Safety in Critical Systems," Technical Report RUCS/97/TR/062/A, University of Reading, UK.

[2] Office of Science and Engineering Laboratories (OSEL) 2011 Annual Report.

[3] Nathaniel Ozarin (2004) Failure Modes and Effects Analysis During Design of Computer Software. Proceedings of The Annual Reliability and maintainability Symposium.

[4] MIL-STD-1543 (1974), Reliability Program Requirements for Space and Missile Systems.

[5] SAMSO-STD 77-2, (1977) Failure Modes and Effects Analysis For Satellite, Launch Vehicle and Re-entry Systems.

[6] R.T. Anderson,(1976) Reliability Design Handbook, IIT Research Institute, Catalog No. RDH-376..

[7] P. L. Goddard (1993)"Validating the safety of real time control systems using FMEA', Proc. Ann. Reliability and Maintainability Symp. pp. 227-230.

[8] P.L. Goddard,(2000) "Software FMEA Techniques", Proc. Ann. Reliability and Maintainability Symp, pp. 118-123.

[9] Goble, W. (2012). The FMEA method .INTECH, 59(2), 14-16,18,20. Retrieved from http://search.proquest.com.ezproxy.libproxy.db.erau.edu/docview/1008351257?accountid=27203

[10] McKinney, B. T. (1991). FMECA, the right way.In Reliability and Maintainability Symposium, 1991.Proceedings., Annual (pp. 253-259) IEEE.

[11] De Miguel, M. A., Fernandez, J., Pauly, B., & Person, T. (2005). Model-Based integration of safety analysis and reliable software development. In Object-Oriented Real-Time Dependable Systems, 10th IEEE International Workshop on (pp. 312-319) IEEE.

[12] Franceschini, F., &Galetto, M. (2001). A new approach for evaluation of risk priorities of failure modes in FMEA.International Journal of Production Research, 39(13), 2991-3002.

[13] Jenab, K., &Dhillon, B.S. (2005). Group-based failure effects analysis (GFEA). International Journal of Reliability, Quality and Safety Engineering, 12(4), 291-307.

[14] Illiashenko, O., &Babeshko, E. (2012). Choosing FMECA-based techniques and tools for safety analysis of critical systems.Information& Security, 28(2), 275-285.

[15] Bozzano, M., &Villafiorita, A. (2010). Design and safety assessment of critical systems.CRC Press.

[16] Haider, A. A., &Nadeem, A. (2013). A survey of safety analysis techniques for safety critical systems.International Journal of Future Computer and Communication, 2(2), 134-137. doi:10.7763/IJFCC.2013.V2.137

[17] Chao, L. P., & Ishii, K. (2007). Design process error proofing: failure modes and effects analysis of the design process. Journal of Mechanical Design, 129(5), 491-501.

[18] Dale, C., & Anderson, T. (2009). Safety-Critical Systems: Problems, Process and Practice: Proceedings of the Seventeenth Safety-Critical Systems Symposium Brighton, UK, Springer Science & Business Media.

[19] K. Jenab and J. Pineau / Management Science Letters 5 (2015), "Failure mode and effect analysis on safety critical components of space travel".pp. 669–678.

[20] J. B. Bowles and C. Wan, "Software failure modes and effects analysis for a small embedded control system," Reliability and Maintainability Symposium, 2001. Proceedings. Annual, Philadelphia, PA, 2001, pp. 1-6.