International Journal of Computer Applications (0975 - 8887)
Volume 143 - No.5, June 2016

Software Safety Analysis of Ball Position Control System using SFMEA

Kadupukotla Satish Kumar
Dept of Computer Science and Engineering
JNTU Kakinada

Panchumarthy Seetha Ramaiah
Dept of Computer Science and Systems Engineering
Andhra University, Visakhapatnam

ABSTRACT

Software Failure Modes and Effects Analysis (SFMEA) is a traditional system safety analysis technique which is widely used in the aerospace, automotive and other safety-critical intensive systems. However, traditional FMEA methods are difficult to identify and analyzing the failure modes which 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 using an experiment for safety-critical embedded control systems. The work presented here provides a general example illustrating how SFMEA can be effectively applied to an 8-bit micro-controller (Chip 89S52) based computer control system having little or no hardware protection. This paper also describes Functional FMEA, interface FMEA, and detailed software FMEAs. The experimental results of SFMEA also found the hardware failures and memory faults. The safety analysis reveals several design deficiencies and physical faults for which modifications are needed. This paper also found 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.

Keywords

BPCS, Software Safety,SFMEA

1. INTRODUCTION

Software plays the primary and central role in the functioning of various kinds of systems like real-time computer control systems, embedded computer systems, etc. In fact, software is increasingly being used to handle safety-critical computer systems which were formerly controlled by humans or hardware. With the increased dependency of systems on software and with its evolving size and complexity, it has now become the major contributor to system failures and a primary threat to system safety and reliability.
The FMEA methodology is one of the risk analysis techniques recommended by international standards. It is a systematic process to identify possible failures existed in an embedded computer system. FMEA is mostly adapted for equipment and material failures, but in a broad sense, human error, performance and software errors can also be included. By applying the SFMEA methodology during the various phases of a product's lifecycle, the methodology provides a systematic strategy for examining all the ways in which a product can fail. The results of SFMEA in turn affect the product design, process development.
Nathaniel Ozarin (2004) [3], FMEA method first distinguishes the failure modes and then analyses the cause and effect of these failure modes. K. Jenab and J. Pineau (2015) [19], In the practice of a SFMEA, analysts collect 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 evaluation of potential failures and their effects.

—The identification and prioritization of actions that could eliminate the potential failures reduce their chances of occurring or reduce their risks.

—The documentation of these identification, evaluation and corrective activities so that product quality improves over time.

Isaksen (1997) described in a technical report as, SFMEA is a bottom-up safety analysis of potential failure modes within a computer system and assessment of the associated effects on system functionality [1]. It is used to identify potential design weaknesses such that they can be mitigated in the early stages of a design program. Weaknesses within the software portion of systems are of particular concern. However as software increases in size and complexity it cannot typically be exhaustively verified as a result, latent software faults have become a leading source of safety and quality issues for medical devices. Coming to medical devices, which are using software, according to the *Office of Science and Engineering Laboratories* (OSEL), USA's annual report 2011, over 20% of all medical device recalls in the United States were due to software failures [2]. R.T. Anderson,(1976) [6], MIL-STD-1543 (1974) [4]

and SAMSO-STD 77-2 [5], (1977) Standards and a handbook have been published to direct its orderly application on military projects. The standards give useful insight into why FMEA is employed.
As per Goble.W (2012), FMEA was created in the 1960s as part of the U.S. Minuteman rocket program in order to find and mitigate unanticipated design problems [9]. McKinney postulated that in order for FMEA to be effective, it must be implemented early in development so that design may be altered to mitigate or eliminate the catastrophic, critical, and safety related failure possibilities [10]. Jenab, K., (2005) developed a group based FMEA which tried to resolve conflict among experts [13][19]. Chao and Ishii (2007) presented an example and case study using a modified FMEA design process [17]. It decomposed the design process into six potential problem areas and used a question-based FMEA approach. Dale (2009) [18] wrote a book which defines safety-critical systems and the processes which can be used to solve issues related to safety. Bozzano (2010) [15] contributed to the safety assessment of critical systems with a book focused on techniques and methods for dependability, reliability, and safety assessment [15]. Illiashenko (2012) [14] declared that there is no universally valid approach for determining which technique to use for reliability analysis. Their study had two main goals; reduce the risk of incorrect safety assessment, and examine FMEA-based techniques to determine how and when to use them for particular tasks. De Miguel (2005) and Franceschini.F (2001) and Haider, A. (2013) stated that use of only one analysis technique is insufficient, and suggest combined usage of methods is important for safety analysis of critical systems [11] [16]. Early researches described limited experimental concepts. They described more theoretical concepts. This paper describes an experimental safety analysis using SFMEA for Safety-Critical Computer Systems. 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 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.

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

The objective of this research work is to enlighten the use of SFMEA for an embedded control system through the development of an experiment. P. L. Goddard (1993), (2000) discussed the approach for SFMEA together with recognizing the types of variables and their failure modes [7] [8]. Experimentation of the proposed methodology is based on the ball position controller system, which is shown in figure 1. The control objective of this work is to regulate the flow of air into a plastic tube so as to keep a small light weight ball suspended at a predetermined height called the set-point. Increasing the flow raises the ball and decreasing the flow lowers it. The BPCS experiment consists of: 3-feet long white plastic tube, light weight ball, DC motor fan, and ultrasonic sensor circuit and 89S52 micro controller.
The vertical 3-feet long clear plastic tube attached to a stand, which contains a light weight ball inside, a DC motor fan at the bottom to lift the ball, and an ultrasonic sensor at the top to sense the balls height. The tube is connected to the DC motor fan inlets via an input manifold which has an inlet at the bottom as indicated. There is an output manifold at the top of the plastic tube with an outlet as shown. The presence of the manifolds is a key part of the experiment. The ultrasonic sensor circuitry detects the position of the light weight ball and the micro-controller regulates the power supply applied to the DC motor fan so as to control the air flow into the white plastic tube, keeping the light weight ball at a predetermined height.

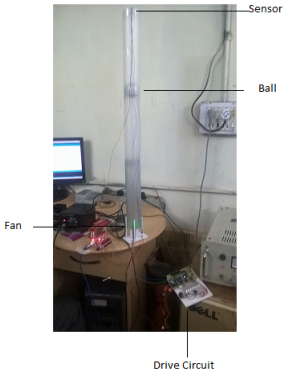

Fig. 1. Ball Position Control System (BPCS)

BPCS Explanation
The light weight ball position control system experiment is a system composed of five modules, where one of them has a DC motor fan to blow air into the white plastic tube moving a Styrofoam light weight ball inside it as shown in figure 1.The modules are instrumented with an ultrasonic sensor to sense the light weight ball height. The air flux coming from the DC motor fan is transferred to the white plastic tube using a pressurization cylinder whose objective is to reduce the turbulence produced by the DC motor fan, which delivers an almost laminar flux. Each module is coupled with the others by the operational objective of the system is to maintain the light weight ball at a predefined height in the plastic tube. The software in the micro-controller provides the control mechanism; it implements a proportional controller that is, the output signal data is proportional to the amount of error in the balls position relative to the set-point. The software is implemented in assembly language, which is assembled and downloaded into the system internal ram of micro-controller. This ensures that the program stays in the micro-controller when the system power is out.
A Block diagram of the BPCS system is shown in figure 2. Each module is coupled with the others by a common manifold. The base box corresponds 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 distributed over each module in a parallel way. Depending on the power applied by the DC motor fan and the input size of the manifold, the air flux continues its trajectory moving the ball inside it. The air from the plastic tube is combined again in the output manifold and ejected through the output, in the right side of the box. This reconfigurable structure possesses input and output manifolds in individual boxes that can be connected between them by their design as a Lego piece. The BPCS dynamical model comprises an energy transfer by airflow from the DC motor fan to the light weight ball. This transfer is typically nonlinear.
The micro-controller generates a Pulse Width Modulation (PWM), it is a modulation technique used to encode a message into a pulsing signal. Although this modulation technique can be used to encode information for transmission, its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. PWM generates a control signal to regulate the

speed of the DC motor fan. The higher the duty cycle of the PWM signal, the faster the fan runs. An output control circuit between the micro-controller and the DC motor fan provides hardware "stops that ensures that the PWM signal is within the specified "safe operating range.

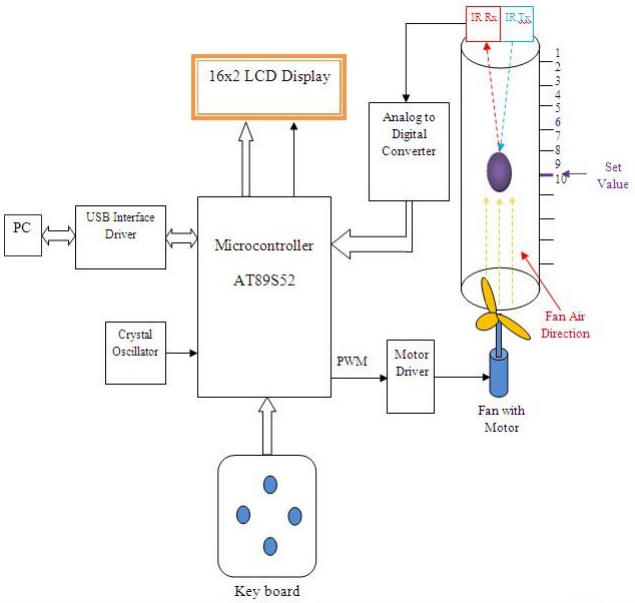

Fig. 2. Block Diagram of Ball Position Control System (BPCS)

If the Pulse Width Modulated cycle is out of the meticulous range, the fault pointer light is switched-on and the system is closed. This fault verification function ensures a safe functional range for the DC-motor even in the occurrence of a software malfunction. The altitude of the ball is determined by the sensor. The ultrasonic sensor generates high-frequency sound waves and evaluates the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance the ball is from the sensor. If the ball is nearer to the sensor, the time interval between the transmission of sound wave and reception of echo will be very small. The output power of the sensor is sample by the converter Analog to Digital, which is embedded in the micro-controller. The inward power is transformed into a one-byte value in an ADC register, which demonstrates elevation 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 deliberated ADC value from the set-point value. The variation is multiplied by a constant to determine the rectification to the PWM signal. If the intensity of the error is high, then more correction is required.
If the calculated ADC value is below the set-point variable, the ball is above the set-point and the micro-controller enhances the duty cycle to increase the speed of the fan and raise the ball. On the other hand, if it is below the set-point, the duty cycle is reduced to lower the ball. The input control circuit sandwiched between the ultrasonic sensor and the micro-controller 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 ultrasonic sensor failures. For example, if the sensor shorts out and outputs an enormously high power (e.g. 10 V) to the micro-controller without a safety control circuit, then the micro-controller 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 serial port and THEN 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 micro-controller triggers an Interrupt Service Routine (ISR) every 5 microseconds: 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 (milliseconds) period with a 50% duty cycle.

3. SFMEA FOR BPCS

In the process of a SFMEA, analysts compile lists of modules failure modes and try to infer the effects of those failure modes on the system. System models, typically simple engineering diagrams, assist analysts in understanding how the local effects of modules failures propagate through complex architectures and ultimately cause hazardous effects at system level. The system has five main functional modules:

(1) Ball-position-controller
(2) DC motor fan and drive circuit
(3) Ultrasonic 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.

Table 1. Functional SFMEA for 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 2. Functional SFMEA for Ball position controller module

Module	Failure mode	Component Effect	Failure impact	Fault Category	Observation
Analog to Digital converter	Failure to get or convert position (in hardware)	Incorrect output data	DC motor fan speed does not change	D	Setup hardware or software checking to ensure the input data is updated periodically
	Too much delay in getting value	Response data too slow	DC motor fan speed is incorrect	D	Set up software checking to measure the pin transition time.
	Incorrect value in interrupt register	Incorrect output data	DC motor fan speed is incorrect	C	Set up software checking to ensure the value within the tolerance

Table 3. Interface FMEA for 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 checking to ensure the input voltage within the tolerance
	P in P 0.1 stuck 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
	P in P 0.1 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 checking to ensure the output voltage within the tolerance

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.
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

Table 4. Detailed SFMEA for output Response Computation

Variables	Failure Modes	Critical Software Variables					
		Set Height(30% duty cycle)	Error (Frequently 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	$\frac{1}{3}$	40	50	$\frac{100}{256}$
	Value exceeds allowed tolerance low	≤ 20	50 high	$\frac{1}{3}$	40	50	$\frac{100}{256}$
Error	Value exceeds allowed tolerance high	30	$FF (8 bits) or a big positive number	$\frac{1}{3}$	40	50	$\frac{100}{256}$
	Value exceeds allowed tolerance low	30	$80 (8 bits) or a big negative number	$\frac{1}{3}$	40	50	$\frac{100}{256}$
Constant for p-Controller	Value exceeds allowed tolerance high	30	50	$FF (8 bits) or a big positive number	40	50	$\frac{100}{256}$
	Value exceeds allowed tolerance low	30	50 high	$80 (8 bits) or a big negative number	40	50	$\frac{100}{256}$
Previous duty cycle	Value exceeds allowed tolerance high	30	50	$\frac{1}{3}$	≥ 70	50	$\frac{100}{256}$
	Value exceeds allowed tolerance low	30	50	$\frac{1}{3}$	≤ 20	50	$\frac{100}{256}$
Present duty cycle after updating	Value exceeds allowed tolerance high	30	50	$\frac{1}{3}$	40	50	$FF (8 bits) or a big positive number
	Value exceeds allowed tolerance low	30	50 high	$\frac{1}{3}$	40	50	$80 (8 bits) or a big negative number
Constant for converting the scale	Value exceeds allowed tolerance high	30	50	$\frac{1}{3}$	40	50	$\frac{100}{256}$
	Value exceeds allowed tolerance low	30	50	$\frac{1}{3}$	40	≤ 20	$\frac{100}{256}$

fault category. The observation section recommends installing is redundant sensor to detect the balls location and restart the system.

3.1 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 micro-controller 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 analog to digital converter. Table 2 display failure modes, component effect, failure impact on the system and fault category.

3.2 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.

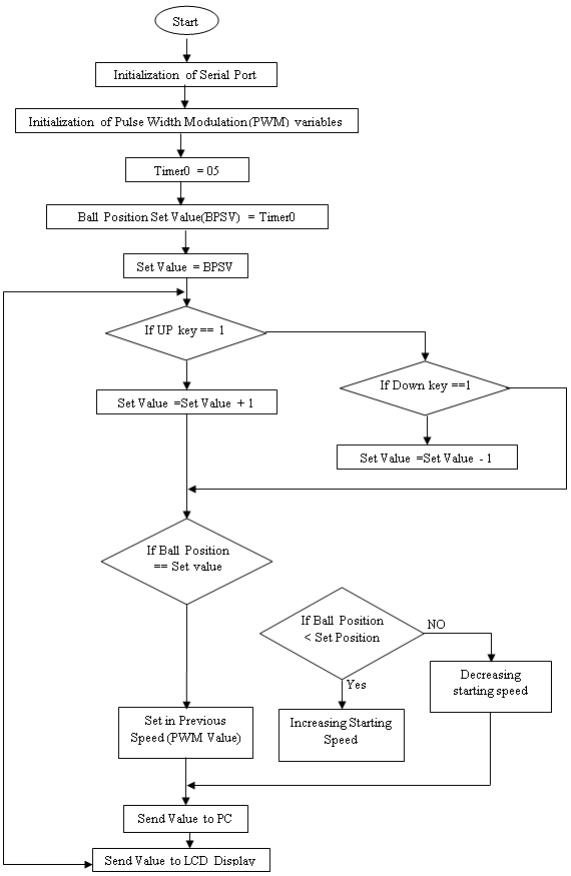

Fig. 3. Flow chart for the main program of the BPCS

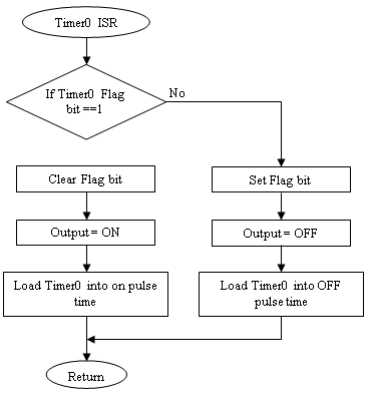

Fig. 4. Flow chart for the ISR of the BPCS

An experiment was carried out of the BPCS. It was assumed that the hardware of the BPCS is fault free. An experiment comprising of 100 trial runs of the system where every trial run comprised of 30 minutes. The possible failures of the interface between the ISR and the generation of the PWM signal were observed as shown in Table 3. Consider the first failure mode Data register holds the intended duty cycle for the PWM signal; if its value is erroneously high, the DC motor fan blows at too high a speed. This results in the light weight ball shooting to the top of the plastic tube. One corrective action to take in response to this failure is to set up a hardware or software check to ensure that the balls rise is within a certain height tolerance of the set-point. If it rises more than the tolerance value above the set-point, an error should be signaled to the operator. Observe that this failure mode has the same effects as failure mode in Table 1; thus it is assigned to the same fault category.

3.3 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.

4. CONCLUSION

SFMEA is an inductive, 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 micro-controller 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 finds 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 balls 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.

5. REFERENCES

[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. InReliability 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. 669678.