Proceedings of the 2016 IEEE/SICE International Symposium on System Integration, Sapporo Convention Center, Sapporo, Japan, December 13-15, 2016

A Study of Methodology for Securing Control Software based FMEA-FTA Coordination

Masakazu Takahashi, Riki Kosaka, Reiji Nanba, Yunarso Anang, and Yoshimichi Watanabe

Abstract— **Many industrial products are controlled by software. Errors in the control software make the products and users danger. To avoid this situation, it is necessary that inexpected behaviors and operations do not make the products unsafe state. This paper proposes a method that the control software makes safe by conducting "Failure Mode and Effects Analysis (FMEA)" and "Fault Tree Analysis (FTA)" repeatedly. The outline of the proposed method is as follows. In the upper phase, risks of control software are analyzed by using FMEA exhaustively, and the measures are reflected to the specifications. In the lower phase, risks that cannot be taken the measures are clarified, and the measures are reflected to the specifications and software. FMEA and FTA are conducted repeatedly, until the control software does not contain risk.**

I. INTRODUCTION

Computers have been installed into industrial products, while the use of software has become popular for controlling such products. The kind of software used to control industrial devices and products is referred to as control software. Recently, trouble and problems in industrial products due to unsafe factors derived from control software have been increasing [1, 2, 3]. Unsafe factors existing in control software can have significant impact on human life and industrial products. Therefore, methodology for securing control software safety has become a requirement.

In this paper, we propose a method for securing control software safety through the entire development process. The following section outlines the proposed method. In the upper process of development, "Failure Mode and Effects Analysis (FMEA)" is applied to the requirement specification and the functional specification of control software in order to exhaustively clarify unsafe factors inherent in control software. Standard measures for securing software safety are proposed, and risks are reduced to the allowable degree by conducting those measures. In the lower process, "Fault Tree Analysis (FTA)" is applied to developed control software in order to clarify the causes of those unsafe factors which are not addressed in the upper process. Risks caused because of unsafe factors are reduced to the allowable degree by revising the design specification and control software. After control software has been developed, FMEA and FTA are further conducted repeatedly in order to clarify new unsafe factors.

M. Takahashi and Y. Watanabe are with the Univ. of Yamanashi, graduate school dept. Div. of Eng. Sect. of Information Eng., Takeda3-5-11, Kofu, Yamanashi, 400-8511 Japan (phone: +81-55-220-{8585, 8651}, e-mail: {mtakahashi, nabe}@yamanashi.ac.jp).

R. Kosaka and Y. Anang was a M.S. and Ph.D student at Univ. of Yamanashi, , Takeda3-5-11, Kofu, Yamanashi, 400-8511 Japan (e-mail: {g15mk007, g14dma01}@yamanashi.ac.jp).

R. Nanba is with the Daiichi Institute of Technology, Faculty of Eng, Kokubu-Chuou 1-10-2, Kirishima, Kagoshima, 899-4395 Japan (phone: +81-995-45-0640, e-mail: r-nanba@daiichi-koudai.ac.jp).

II. RELATED STUDIES

Related studies are classified into studies regarding the establishment of the standards associated with software safety, studies regarding software FMEA and FTA, and studies regarding the development of safety analysis tools.

First, let us focus on the establishment of standards associated with software safety. In the automotive industry, the functional safety standards, ISO26262, were established [4]. ISO26262 specifies procedures to enhance safety for control software by applying "Hazard and Operability Study (HAZOP") [5], FMEA, and FTA. As for control software used for drug manufacturing facilities, International Society for Pharmaceutical Engineering established the guideline called Good Automated Manufacturing Practice Ver. 5 (GAMP5) [6], and Ministry of Health, Labor and Welfare set the Guideline on Management of Computerized Systems for Making Authorization Holders and Manufacturers of Drugs and Quasi-drugs [7]. However, those guidelines only provide processes for developing highly-safe control software.

Our next focus is studies related to software FMEA and FTA. Takahashi proposed the implementation method of FMEA for control software which is used for manufacturing drugs [8]. Morita discovered program bugs by listing failure modes and estimated causes from blocks [9]. Niwa proposed and implemented reliability improvement measures by listing failure modes based on the external design unit [10]. Goddard conducted FMEA while defining the failure mode in the command level of control software [11,12]. Snooke conducted FMEA by converting software into an equivalent circuit [13]. Next, let us focus on studies related to software FTA. Weber analyzed fault causes by utilizing FTA for aircraft control software [14]. Friedman proposed an automatic fault tree (FT) creation method for malfunction of software [15]. Moreover, Leveson proposed an FTA approach for control software based on a combination of fault tree (FT) templates corresponding to the programming language [16]. Expanding Leveson's approach, Takahashi et al. proposed a method to mechanically create FT for control software [17].

The last focus is regarding safety analysis tools. Those commercially-available FMEA support tools include AutoFMEA of Toyo Corporation [18], and FMEA-Pro of IHS Inc. [19]. When it comes to FTA support tools, PTC's PTC Windchill FTA is one option [20]. However, software is actually outside the scope of these tools. In addition, they cannot be coordinated and interchanged.

Although related studies have been done individually as described above, specific methods for supporting the securing of control software seamlessly throughout all developmental processes have been in the pipeline and awaited.

978-1-5090-3328-7/16/$31.00 ©2016 IEEE

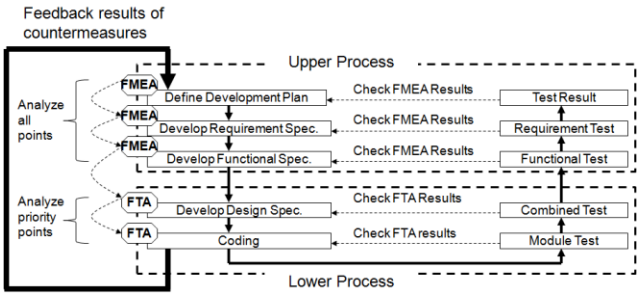


Figure 1. Development Process for safe software

III. Outline of the Proposed Method

This chapter overviews the proposed method for securing software safety. Section A outlines the proposed method. Section B describes the FMEA method for exhaustive analysis of software unsafe factors. Section C and D describes FMEA and FTA method for analyzing the causes. Finally, section E describes the environment for securing software safety based on a combination of FMEA and FTA.

A. Process of Achieving Safe Software

The following section outlines the proposed method. The proposed method secures control software safety by repeating FMEA and FTA. This method also implements safety measures throughout all the processes of control software development. Figure 1 shows methods for securing software safety which are applied in each process of control software development, including the entire flow.

In the upper process, the proposed method examines potential failures and malfunction in the stage of creating the development plan, requirement specification, and functional specification. FMEA is used for this exhaustive examination. FMEA lists failure modes according to each functional part which composes control software. FMEA also clarifies the influence of the failure mode, which occurred in a functional part, on the entire control software system as a failure. In addition, it clarifies measures to be taken and priorities according to the level of influence. Concerning higher-priority failures, the development plan, the requirement specification, and the functional specification are revised. Measures which are accompanied with partial program revision are implemented during the lower process.

Next, program measures which cannot be implemented in the upper process are implemented during the stage of creating the design specification and the coding stage. FTA is conducted for the software faults discovered later in order to clarify the fault causes and revise the design specification and the program. As for all faults, these measures reduce the influence of failure on the entire control software system to the allowable degree.

The first session of examination for securing control software safety is completed through the processes above. However, implementation of a wide variety of safety measures could produce a new failure mode in a functional part of control software, or could cause another fault which affects the entire control software system. for control software after the first examination for safety. Repeating this process until all failures and faults become allowable levels can achieve development of secure control software.

B. Outline of FMEA procedure

This section describes the FMEA method utilized for examining the safety of control software in the upper process.

FMEA divides the target industrial product into components in order to list physical faults of these components. At this time, it is assumed that the target industrial product and its components are in proper condition. Faults of these components are referred to as failure modes. The influence (failure) of a failure mode occurred given to the entire product system is then examined. Clarifying failures due to all failure modes and implementing proper measures can secure safety for the industrial product. Program bugs are generated during the stage of software creation, so that the software program itself might not be in proper condition. Therefore, program bugs are not referred to as failure modes. Control software program bugs can adequately be removed by testing. Based on this concept, we decided to exclude program bugs from FMEA targets. Therefore, we set the following deviations as control software failure modes handled in this study: Deviation of appropriate use of control software components maintained in proper condition by testing, and deviation of operational procedure. The proposed method analyzes the results of FMEA for the existing control software and clarifies common failure modes and standard measures.

Figure 2 shows the flow of the proposed FMEA. First, functionas of control software are listed based on requirement specifications and functional specifications as input. Second, usage and operational procedures are confirmed whether they correspond or not according to each functional unit listed. Third, when they correspond, corresponding common failure modes are clarified. Additionally, a correspondence table is created. Fourth, functions, common failure modes, and influences on the control software are identified in the FMEA sheet. Fifth, Severity, Incidence, and Detection Rate are determined according to each failure mode in order to determine the risk priority (by entering values in each upper columns in the FMEA sheet). Risk evaluation matrices in Figure3 are used for determining the risk priority. By using the left matrix in Figure 3, the risk class is obtained from Severity and Incidence. By using the right matrix in Figure 3, the risk priority is obtained from the risk class and the detection rate. Sixth, based on the risk priority, it is determined whether the control software can tolerate the relevant failure or not. If intolerable, the application of the standard measures in Table 1 is then considered. Finally, Severity, Incidence, and Detection Rate where the standard measures are applied are re-evaluated in order to determine the risk priority (by entering values in each lower columns in the FMEA sheet). Confirmation that the control software can tolerate the relevant failure brings an end. If the failure is not tolerated, other standard measures are considered, while the Severity, Incidence, and Detection Rate of the failure is re-evaluated. These processes are repeated until the control software can tolerate all faults.

C. Outline of FTA procedure

This section describes the FTA method utilized for examining the safety of control software in the lower process.

FTA traces the causes which cause particular unfavorable phenomena (faults) of the target product in a phase manner in order to clarify them. When tracing the factors , FTA focuses

List of Common Failure Mode

Group	Common Failure Mode	Failure Example	Countermeasure Policy	Representative Risk Reduction Measures
Startup	The startup conditions for functions are not prepared	Related operations cannot be conducted, an improper system status exists	Review the startup conditions	Add the confirmation procedure for the startup conditions to the SOP (3), set the conditions as
			Conduct multiple checks when	
			Conduct the startup check	Display the startup status (4)
Termination	The termination conditions for functions are not prepared	Related operations cannot be conducted, an improper system status exists	Review the termination	Add the confirmation procedure for termination conditions to the SOP, set the conditions whether or not to terminate (4), multiplex the
			Conduct multiple checks upon termination	
			Conduct termination check	Display the termination status (4)
			Transit to the safe status for	Add the emergency stop function (4)

refer & use

Function vs. Failure mode Correspondence Table

Function1,
Function2,
- - - -

input

Requirement Spec./
Functional Spec.

RequirementSpec. / Functional Spec.	Gropu	Common Falure Modes
Function A	startup	The startup conditions for the function are not prepared.
	termination	The termination conditions for the function are not prepared.
	input/output	Instruction on SOP missed.
Function B	startup	The startup conditions for the function are not prepared.
	input/output	Instruction on SOP missed.
	input/output	Mistake in checking product.
- - - -	- - - -	- - - -

Analyze correspondences between functions and common failure modes considering similarity.

develop

Result of Failure Mode and Effects Analysis

Function	Common Failure Mode	Impact to System	Accept /Reject	Severity	Incidence	Risk Class	Detection rate	Priority	Measures
Function A	Startup condition X is not prepared.	The machine does not	○	Middle	Low	3	High	Low	Add SOP for Checking Startup
				Middle	Low	3	High	Low	
	Termination condition Y is not prepared.	The machine use electric power continously.	○	Middle	Low	3	High	Low	Add SOP for cheking termination Condition.
				Middle	Low	3	High	Low	
- - - -	- - - -	- - - -	--	--	--	--	--	--	- - - -

Figure 2. Flow of Proposed FMEA Procedure

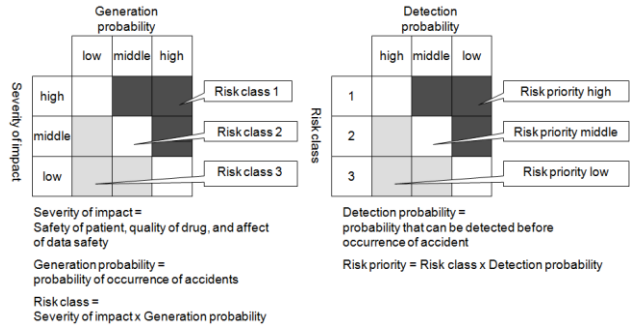


Figure 3. Risk Assessment Matrix

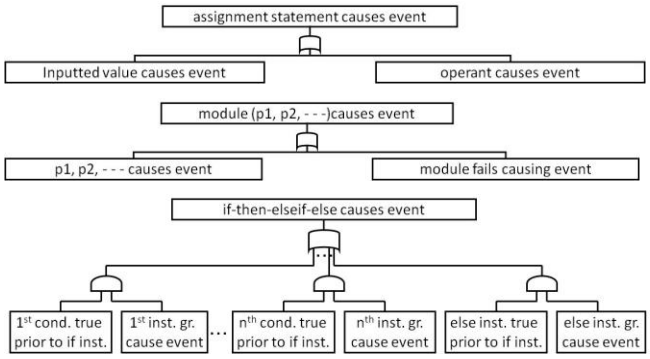


Figure 4. FT templates for Control Software

on the product's components and the relationships between the components. Those paths (causal relationships) of influences that lead to the causes from the fault are expressed in a tree structure with logic symbols and event symbols. This tree is referred to as a Fault Tree (FT). Control software is composed of basic commands as components, while execution sequences of these basic commands are interfaces between the components. In the proposed method, while analyzing the existing control software written in C, we created FTA templates for control software. The following seven types of template were created: assignment, if-then-else, while, module call, interrupt, statement inexecutable, and global variable. Figure 4 shows assignment, module call, and if-then-else FT template. As for the interfaces between components, FT templates are connected while the execution sequences of commands are traced from the one where a fault occurs. FT templates are then connected until the execution sequence of commands can no longer be traced. The phenomenon existing in the node on the edge of the FT is finally determined to be the cause of the fault. Figure 5 shows the FTA procedure. Note that we used program slicing in order to trace the execution sequence of basic commands. Program slicing is a technique that clarifies dependencies between commands within the program. This technique extracts all the commands that could affect the execution of the commands.

D. Safety Analysis Support Environment

This section describes a safety analysis support environment achieved by integrating all the processes and methods described in section 3.A through section 3.C.

Figure 6 shows the outline of the safety analysis support environment. This environment consists of the FMEA/FTA support tool, and safety information management database.

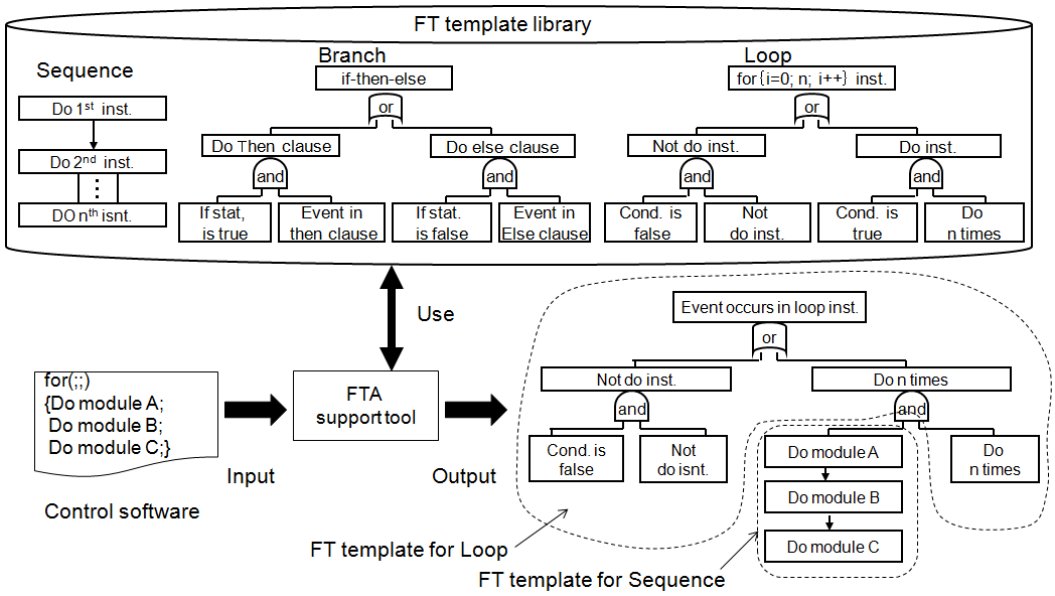


Figure 5. Flow of Proposed FTA Procedur

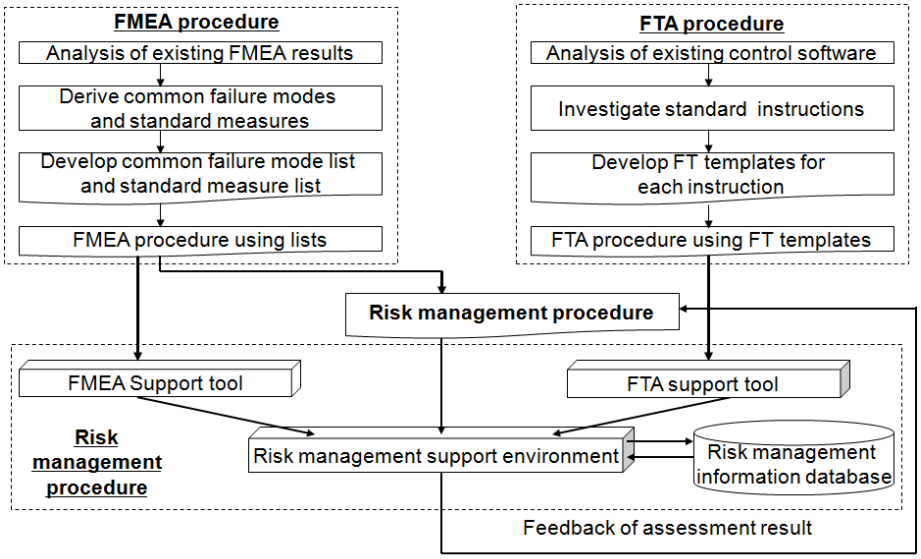


Figure 6. Proposed safety support environment

Function	Failure Mode	Influence on the System	Adoption	Function Importance	Occurrence Rate	Detection Rate	Risk Priority	Countermeasures
Manufacturing instruction data read	Function startup conditions are not prepared	Measurement interrupted	○	Middle	Middle	High	Low	Add the confirmation procedure for startup conditions (URS and SOP), RT
				Middle	Middle	High	Low	
Manufacturing instruction data read	Function startup conditions are not prepared	Measurement interrupted	○	Middle	Middle	High	Low	Add the confirmation procedure for termination conditions (URS and SOP), RT
				Middle	Middle	High	Low	
Manufacturing instruction data read	SOP is misread, HMI is misread, Mistake in checking products	Improper measurement, drugs with improper quality are manufactured	○	High	Middle	High	Middle	Conduct multiple checks on instruction data (URS and SOP), RT,
				High	Low	High	Low	Perform a comparison with the screen (URS and SOP), RT
Manufacturing instruction data read	Inputting errors	Drugs with improper quality are manufactured	○	High	Middle	High	Middle	Perform a comparison of instructions (URS and SOP), RT,
				High	Low	High	Low	Compare the measurement results and instructions (URS and SOP), RT
Manufacturing instruction data read	The lower limit of calculation precision is confirmed	Improper measurement, drugs with improper quality are manufactured	○	Middle	Low	High	Low	Add the warning function for a small divisor (URS, FS, DS, and MS), (RT, FT, IT, and MT)
				Middle	Low	High	Low	
Manufacturing instruction data read	Long time intervals for function calibration	Errors produced in measurement	○	High	Middle	High	Middle	Calibrate devices before use (URS and SOP), RT
				High	Low	High	Low	
Manufacturing instruction data read	Wrong operation authority	Drugs with an improper quality are manufactured	○	High	Low	High	Low	Confirm qualifications before operation (URS and SOP), RT
				High	Low	High	Low	
Manufacturing instruction data read	Backup failure	Data loss	○	High	Low	High	Low	Conduct backup immediately after data read (URS and SOP), RT
				High	Low	High	Low	
Manufacturing instruction data read	Reception of data with virus attached	Improper measurement	○	High	Low	High	Low	Conduct virus check on data to be input in advance (URS and SOP), RT
				High	Low	High	Low	

Figure 7. Output of FMEA support tool

The FMEA support tool is used in the creation stage of the development plans, the requirement specification, and the functional specification. A list of functions extracted from the requirement specification and the functional specification is used as input to the FMEA support tool. Output is the FMEA results. Figure 7 shows the output results of the this tool. As the output is written as comma-separated-value file, Figure 7 is expressed using table format for readability.

The FTA support tool is used in the creation stage of the design specifications and the program. Faults and control software are input to the FTA support tool. FT is the output for faults. Figure 8 shows the output results of this tool. As the output of the FTA is written as comma-separated-value file, Figure 8 is used indent style for readability.

The safety information database manages data necessary for conducting FMEA and FTA. This database consists of the seven tables. The requirement specification and functional specification table have information regarding functions contained in the specifications. The failure mode table has information regarding all common failure modes. The correspondence table has information regarding relationships between functions and common failure modes. The failure/fault table has information regarding the direction of failure measure according to each common failure mode. The actual measure table has information regarding specific measures, and the fault tree table has information regarding the fault tree according to each fault. When new information is obtained, such new information is amassed by implementing the Plan-Do-Check-Action cycle.

IV. EVALUATION

In order to evaluate the proposed method and support environment, we applied them to the development of control software for an support device that is a seating chair that helps elderly persons to stand up. Figure 9 outlines this support tool. This support tool consists of the mechanism part and the control part. The mechanism part is used by placing on the chair seat. This mechanism part has a structure with two hinged aluminum panels between which a balloon is inserted. Blowing and shrinking this balloon adjusts the seating angle of the top aluminum panel. This action enables elderly persons who are seated to lean their upper bodies forward (a posture that enables a person to stand up easily) to help them stand up. The control part consists of the controller (Lenovo Thinkpad E430C) and various sensors. The sensors are the acceleration sensor (KXM52-1050 of Kionix), which measures the acceleration of the vertical movement when elderly persons stand up, and the pressure sensor (FSR#402 of INTERLINK ELECTRONICS), which measures arm force. We used Gainer I/O modules for connecting the computer and the sensors. Processing language was adopted as the programming language.

By the way, we developed the FT templates used by the proposed method while assuming C language to be used, not processing language. However, we only used simple commands (assignment statement, if-then-else statement, while statement, module call, global variable) for the control software. Therefore, we were able to use developed FT templates, and we did not need to develop FT templates for processing language.

```
PERIOD too high
AND
|→TL2 too small
|   (detail of this branch is omitted)
|
|→TL1 = SAMPLE(L1)
    OR
    | → SAMPLE(L1) is executed
    |     | → fault in SAMPLE(L1) => (final event)
    | → SAMPLE(L1) is not executed
          OR
          | → IN CASE1
          |     AND
          |     | → WDTCR mod 16 = 1
          |     | → WDCTR := WDTCR + 1
          |           OR
          |           | → RESTART4 is executed
          |           |     | → TELEMTRY interrupt is occurred 1 time (final event)
          |           | → RESTART4 is not executed
          |                 (detail of this branch is omitted)
          | → IN CASE2
          |     (detail of this branch is same as CASE1)

    (continued)
```

Figure 8. output of FTA support tool

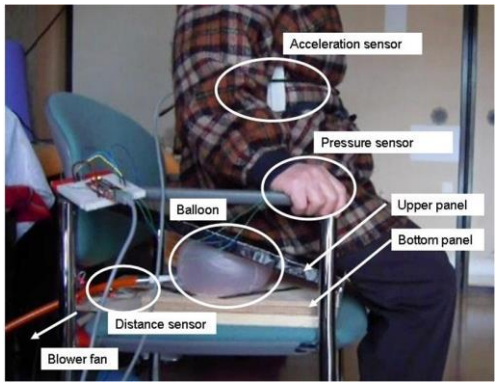
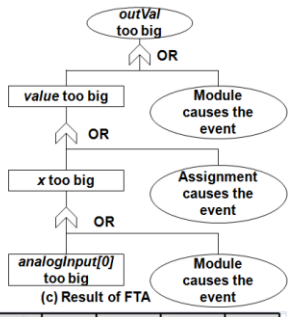


Figure 9. Standing-up support tool for elderly person

```
import processing.gainer.*
void setup {
    :
  gainer.beginAnalogInput();
    :
}
void draw {
    background(0);
    :
(1) float x =
    9.8 * map(gainer.analogInput[0], 85, 212, -1,1);
    :
(2) float value = ---- ;/* decaide x */
(3) int outVal = round(value);
(4) gainer.gigitzlOutput(outVal);
    :
}
```

(a) outline of control software



(c) Result of FTA

Function	Common Failure Mode	Impact to System	Accept/ Reject	Severity	Incidence	Detection Rate	Priority
Seat Angle Control	Start up conditions for leaning are not prepared.	Support tool does not work.	Accept	Low	Middle	High	Low
	Termination condition for leaning are not prepared.	Support tool leans too much	Reject	High	Middle	High	High
- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -

(b) Result of FMEA

Figure 10. Application result of the proposed method

Here, the result that the proposed method and support environment were applied to the control software is described below. In the requirement definition phase, an event, the seat becomes steep slope because the seat angle control function of the control software does not work appropriately, was found by using FMEA, and the event was analyzed by using FTA after completion of the control software. FTA for whole of the support tool was conducting before conducting FTA. As a result, we found that the event occurred when the variable *outValue* in the control software became too big. Figure10(a)

shows the outline of the control software, Figure10(b) shows the result of FMEA, and Figure10(c) shows the result of FTA for the event "*outValue* is too big". The structure of the control software is explained as followings. The function "setup" declares continuous input of the data of seat angle, and the function "draw" inputs seat angle data and outputs control signal to blower fan for expanding the balloon. When the control function of blower fan does not stop, the balloon continues to expand. Consequently, the seat angle becomes steep slope or the explosion of the balloon occurs. The measures are necessary to avoid this serious situation. As for the event, FTA is conducting after the completion of control software. The event is set as a top event, and the event is caused in line (4) of the control software in Figure 10(a). The *outValue* is converted into integer type in line(3). The FT template for module is applied because type conversion function is considered as module. It is assumed that the type conversion function does not have any errors, because it is a standard equipped function. The other functions are considered as same. As a result, the cause is considered that the input to *value* in line (3) is too big. *Value* is calculated in the formula in line(2) (the detail of the formula is omitted). The FT template for assignment is applied. As a result, the cause is considered that the input to "*x* is too big". *X* is inputted in line (1). The FT template for module is applied because the function map is considered as module. As a result, the cause is considered that the "*analogInput[0]* is too big". FTA is finished because the further tracking of the cause cannot be conducted. As a result of FTA, the original cause that *analogInput[0]* is too big is considered as a failure of the sensor, because the software does not contain any failure. We add following functions as measures; the function that *outValue* is not outputted when *analogInput[0]* is too big, and the function that *outValue* is not outputted when the integrated value of *outValue* exceeds the threshold. Those measures make the control software safe. Still, in the actual safety analysis for support tool system, we proposed measures to the hardware of the tool based on the results of the proposed method, such as redundancy of the sensors, and addition of preventing equipment for too much leaning of the seat.

V. Future Issues

In this paper, we proposed a method to secure control software safety by applying FMEA and FTA repeatedly. Our proposal also included a support environment for the proposed method. Our attempt to apply the proposed method and support environment to the development of control software for a support tool to help elderly persons stand up clarified effective measures for securing control software safety. This confirmed that our proposed method and support environment could work effectively.

In the future, we want to apply this proposed method and support environment to more control software. Based on the results, we will improve the proposed method and support environment. In particular, we will try to enhance common failure modes and FT templates. Currently, the results of FMEA and FTA are output in Comma-Separated-Value type files, which give us lower readability. Therefore, we will also examine development of interfaces that enable engineers to understand the FMEA and FTA results intuitively.

References

[1] TOYOTA motor corporation, 75 years of TOYOTA, Voluntary Remedies of Recalls, http://www.toyota-global.com/company/history_of_toyota/75years/text/leaping_forward_as_a_global_corporation/chapter5/section3/item1.html (2016.8.1 accessed).

[2] SEBoK, Medical Radiation Case Study, http://sebokwiki.org/wiki/Medical_Radiation_Case_Study (2016.8.1 accessed).

[3] Japan Aerospace Exploration Agency, Operation plan of X-ray Astronomy Satellite ASTRO-H(Hitomi), http://global.jaxa.jp/press/2016/04/20160428_hitomi.html (2016.8.1 accessed).

[4] Japan Automobile Research Institute, Functional Safety, http://www.jari. or.jp/tabid/223/Default.aspx (2016.8.1 accessed)

[5] HAZOP and Plant Safety Promotion , An elementary Knowledge of HAZOP, http://hazop.jp/hazop_basic.html (2016.8.1 accessed).

[6] GAMP forum, GAMP5 Risk-Based approach to Complaint GxP Computerized Systems, International Society of Pharmaceutical Engineers, 2008.

[7] Ministry of Health, Labor and Welfare, Guideline on Management of Computerized Systems for Making Authorization Holders and Manufactures of Drugs and Quasi-drugs, http://members3.jcom.home.ne.jp/yrq01133/computer_guideline/20101021_1021-11.pdf (2016.8.1 accessed).

[8] M. Takahashi, R. Nanba, and Y. Fukue, A Proposal of Operational Risk Management Method Using FMEA for Drug Manufacturing Computerized System, Transaction on The Society of Instrument and Control Engineers, Vol.45, No. 5. pp285-294, 2012 (in Japanese).

[9] M. Morita, Reduction of software bugs using FMEA, Software Quality Management Catalog, Union of Japanese Scientists and Engineers, pp.461-486, 1990 (in Japanese).

[10] M. Niwa, Improvement of Reliability for System design Using FMEA, Software Quality Management Catalog, Union of Japanese Scientists and Engineers, pp467-475, 1990 (in Japanese).

[11] P. L. Goddard. Validating The Safty of Embedded Real-Time Control Systems Using FMEA, Proc. of Annual Reliability and Maintainability Symposium, pp.227-230, 1993.

[12] P. L. Goddard: Software Safty Techniques, Proc. of Annual Reliability and Maintainability Symposium2000, pp.119-123, 1993.

[13] N. Snooke, Model-based Failure Modes and Eff ects analysis of Software, Proc. of 15th International Workshop on the Principles of Diagnosis, pp. 221-226, 2004.

[14] W. Weber, Heidemarie Tondok, and Michael Bachmayer: Enhancing Software Safety by Fault Trees: Experiences from an Application to Flight Critical SW, Proc. of SAFECOMP2003, LNCS 2788, pp.289-302, 2003.

[15] M. A. Friedman, Automated Software Fault Tree Analysis for Pascal Program, Proc. of Annual Reliability and Maintainability Symposium, pp.458-461, 1993.

[16] N. G. Leveson and P. R, Harvey: Analyzing Software Safety, IEEE Transaction on Software Engineering, Vol. 9, No.5, pp.569-579, 1983.

[17] M. Takahashi, Riki Kosaka, and Reiji Nanba: A Study of Fault Tree Analysis for Control Program in Space System, Proc. of 2015 IEEE/SICE International Symposium on System Integration,pp.301-306, 2015.

[18] Toyo Corporation, AutoFMEA, http://www.kumikomi.net/archives/2011/04/esec2011_3.php?page=8(2016.8.1 accessed).

[19] IHS Inc., FMEA-Pro, https://www.ihs.com/Info/ehss/fmea-pro.html (2016.8.1 accessed).

[20] PTC, Windchill FTA, http://www.ptc.com/product-lifecycle-management/ windchill/product-risk-and-reliability(2016.8.1 accessed).