Peertechz

Trends in Computer Science and Information Technology

mail.computerscience@peertechz.com

ENGINEERING GROUP

TRENDS IN Computer Science and Information Technology

ISSN: 2641-3086 DOI: https://dx.doi.org/10.17352/tcsit

Research Article

A Hazard Analysis Method for Embedded Control Software with STPA

Masakazu Takahashi[1]*, Yunarso Anang[2] and Yoshimich Watanabe[3]

[1]Department of Computer Science and Engineering, University of Yamanashi, Japan

[2]Department of Computational Statistics, Politeknik Statistika STIS, Indonesia

[3]Department of Computer Science and Engineering, University of Yamanashi, Japan

Received: 09 November, 2020
Accepted: 19 November, 2020
Published: 23 November, 2020

***Corresponding author:** Masakazu Takahashi, Department of Computer Science and Engineering, University of Yamanashi, Japan, Tel: +81-55-220-8585; Fax: +81-55-2208776; E-mail: mtakahashi@yamanashi.ac.jp

ORCID: https://orcid.org/0000-0003-1778-9119

https://www.peertechz.com

Check for updates

Abstract

This paper proposes an analysis method for hazards that are occurred by interactions between hardware and software when using an apparatus installed an Embedded Control Software (EBSW). Hazard means a state that negatively affects the apparatus when some bad conditions are satisfied. Especially, the purpose of the method is clarifying the EBSW portions that cause the hazards. The outline of the proposed method is as follows; (1)Develop EBSW specifications written in Unified Modeling Language (UML) and accident information, (2) Conduct safety analysis (System-theoretic Process Analysis: STPA) by inputting EBSW specifications and accident information, and generate the list of hazards and hazard scenarios, (3) Develop sequence diagrams corresponding to the hazard scenarios, and clarify program portions (Hazard Causal Factor: HCF) that are causes of the hazards, and (4) Conduct Failure Mode and Effects Analysis (FMEA), and apply countermeasures to avoid occurrences of the hazards. As a result of applying this method to the sample EBSW, we can confirm that the safety EBSW is developed.

Introduction

At fast, the technical terms that are used in this paper are explained. The accident means an event that causes the loss of the target system, and the loss means a negative effect on the users, environments, missions, and target system. The hazard means the system's state that negatively affects the target system when some bad conditions are satisfied.

Recently, industrial products, such as cars, medical apparatuses, and aerospace apparatuses, are developed as the systems that are combined the hardware and software, and their configuration of the apparatuses and controls become complex. As a result, unintended accidents occur when using the industrial products. Those accidents occur when hazards that are occurred by interactions between hardware and software when using an apparatus and some negative conditions that cause the accident are satisfied. This accident model is called as Systems-Theoretic Accident Model and Process (STAMP) model. Additionally, based on the STAMP model, the safety analysis method that clarifies hazards and hazard scenarios is called STAMP based Process Analysis (STPA) [1].

This paper proposes a method that clarifies the hazards and proposes safety countermeasures after completing the development of the functional specifications for Embedded Control Software (ECSW). In the proposed method, STPA is conducted by inputting the ECSW system specifications that are consisted use-case diagrams and class diagrams that are written in Unified Modeling Language (UML). As a result of conducting STPA, hazards are clarified, and hazard scenarios are developed. Sequence diagrams corresponding to the hazard-scenarios are developed and the Hazard Causal Factors (HCFs) are clarified. In this case, the reasons of the HCFs are the execution of methods and/or the non-execution of methods in the class. Based on the STAMP model, the safety analysis method that clarifies the hazards and the hazard scenarios is called a System-Theoretic Process Analysis (STPA).

The organization of this paper is explained below. Section 2 describes the related works. Section 3 describes the outline of the proposed method. Section 4 describes the applications and evaluations of the proposed method. And section 5 describes future works.

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

Related works

This section describes the previous studies and STAMP/STPA.

Previous studies: The previous studies classify into the development of standards for safety ECSW in the various industrial products and the safety analysis methods.

At first, the standards to develop safety ECSW in the various industrial products were explained. The accidents for the industrial products that required the safety in high level gave the negative impacts to the human's lives and the environments. The regulatory authorities required the observance of the development processes corresponding to the development standard to the manufacturers. Additionally, the regulatory authorities required enough safety analysis for the industrial products. As for such development processes, for examples, JIS T2304 [2], IEC62394 [3], IEC82304-1 [4] in the medical device domain were established, Good Automated Manufacturing Practice [5] in the pharmaceutical production system domain was established, ISO26262 [6] was established in the automobile domain, and DO-178C [7] and JAXA JMR-001 [8] were established in the aerospace domain. As those standards did not describe the detail of the concrete safety analysis procedures, it often occurred that the additional tasks were required because of the misunderstanding of the standard.

At second, the various safety analysis methods were explained. Takahashi et al. proposed a method that clarified all accidents that might occur and decide the countermeasures to solve them using the Failure Mode and Effects Analysis (FMEA) [9]. Weber et al. proposed a fault detection method for the avionics software written in assembler using the Fault Tree Analysis (FTA) [10]. Leveson et al. showed that the Fault Tree (FT) could be developed by preparing the FT templates corresponding to the essential instructions of the ECSW and combining those FT templates [11,12]. Takahashim, et al. proposed the development rules that developed FT automatically by tracing the process that caused the accident and combining the FT templates [13]. Pai et al. proposed the method that calculated the reliability of the system by inputting the design specifications written in the UML [14]. Though those methods were to clarify the cause of the failure of the component level of the industrial product, the complex failures that arose from the interactions between the components could not be dealt with. For this problem, Leveson et al. proposed the method that could be dealt with the complex failures (accidents) that arose from the interactions between the components. The details of this method were explained in the next section.

Outlines of STAP and STPA

This section describes the STAMP model and STPA [1].

Figure 1 shows the STAMP model. The STAMP model describes that the system consists of the controller, process model, and controlled process. The process model shows the state of the controlled process that the controller supposes. The controller sends Control Actions (CAs) to the controlled process based on the state of the process model, and the controller changes the state of the process model based on the sent CAs. The controlled process transits the inner state based on the received CA, and the controlled process returns the result as the Feedback Data (FBD) to the controller. In the case that the state of the process model matches the state of the controlled process, the system is in the safe state. In the case that the case that the state of the process model does not match the state of the controlled process, the system is in the unsafe state. At that time, hazards occur.

The procedure of STPA is explained as follows. At first, the accidents and hazards of the target system are defined. Additionally, the Safety Constraints (SAs) are defined. At second, the Control Structure Diagrams (CSDs) are developed. Figure 2 shows an example of the CSD. The CSD defines the components (subsystems and apparatuses) that are necessary to realize the SCs and the interactions (CA and FBD) between components. At third, Unsafe CAs (UCAs) are defined. The CAs that is necessary to conduct SCs in the CSD are identified. UCAs are derived by applying "the 4 keywords to identify the UCAs that cause the hazards (such as not providing, providing, too fast/too late, inappropriate execution sequence, too fast/too long)" to the identified CAs. At fourth, the conditions that every UCA causes hazard are clarified. The controllers and the controlled processes related to the each UCA are extracted from the CSD, and the control loop related to the UCA is clarified. UCA in the control loop is applied to the guide word one by one, and it is considered whether the UCA applied the guide word causes the hazard. Figure 3 shows the 11 guide words that cause the HCF in the control loop. In the case that the hazard occurs, the conditions that cause the hazard are clarified. Those conditions are HCFs. Additionally, the scenarios that include the processes from the occurrence of the HCF to the hazard are developed. At last, the countermeasures that do not cause hazard are developed by considering the hazard scenario.

Outline of the proposed method

This section describes the outline of the proposed method.

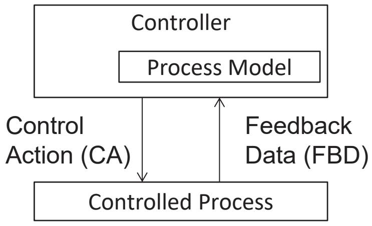

Figure 1: Concept of the STPA model ([14], P.1, fig.1.1-1).

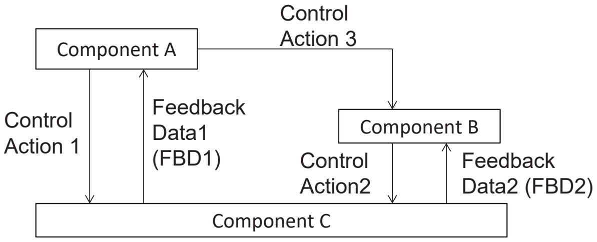

Figure 2: An example of control structure diagram ([14], P.2, Fig.1.1-2).

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

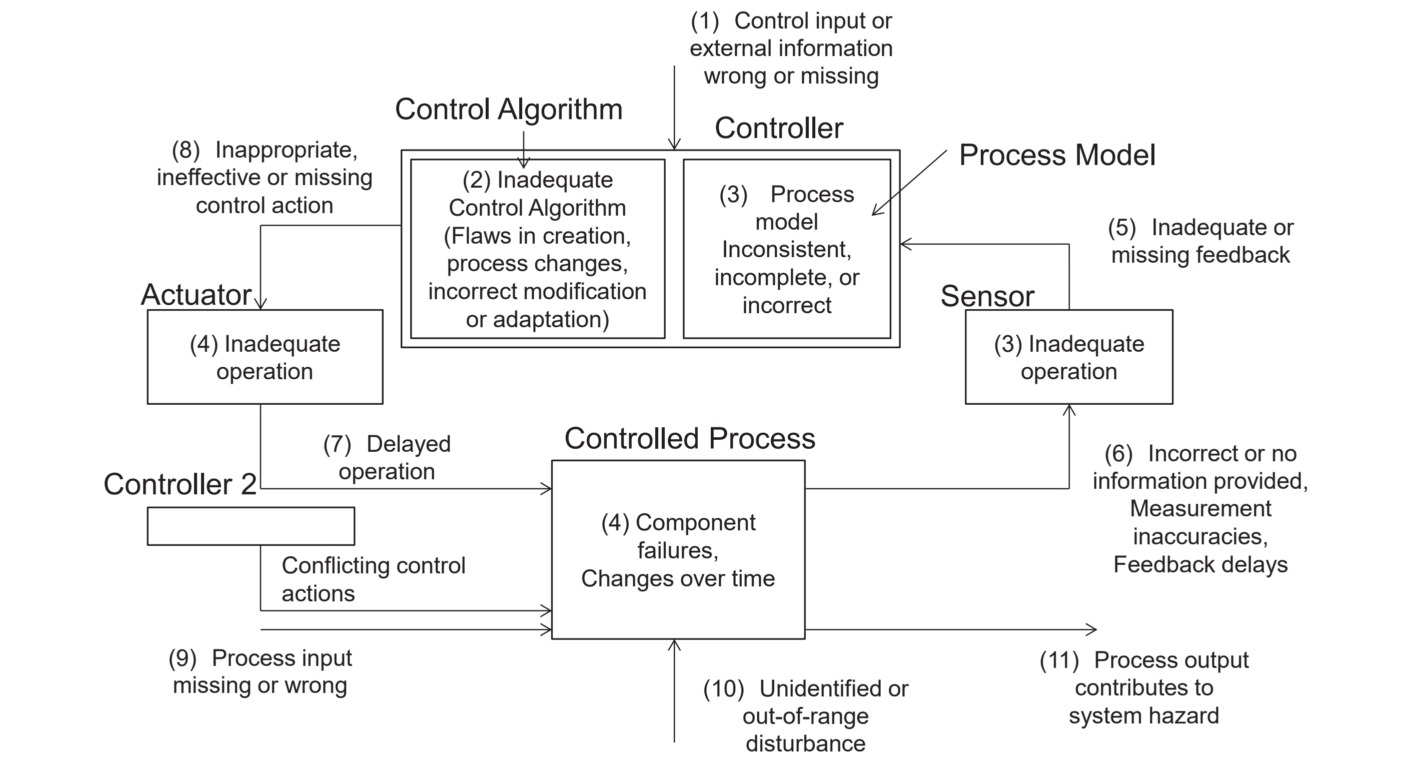
(1) Control input or external information wrong or missing
Control Algorithm
Controller
Process Model
(8) Inappropriate, ineffective or missing control action
(2) Inadequate Control Algorithm (Flaws in creation, process changes, incorrect modification or adaptation)
(3) Process model Inconsistent, incomplete, or incorrect
(5) Inadequate or missing feedback
Actuator
Sensor
(4) Inadequate operation
(3) Inadequate operation
(7) Delayed operation
Controlled Process
Controller 2
(6) Incorrect or no information provided, Measurement inaccuracies, Feedback delays
(4) Component failures, Changes over time
Conflicting control actions
(9) Process input missing or wrong
(10) Unidentified or out-of-range disturbance
(11) Process output contributes to system hazard

Figure 3: "11 guide words" that cause the HCF in the control loop ([14], P.9, Fig.2.5-1).

The subsection A describes the whole outline, and the subsection B describes each task that consists the proposed method.

Outline of the proposed Method

Figure 4 shows the outline of the proposed method. The proposed method can be applied after the completion of the requirement definition and the functional design (completion of the development of the use-case diagrams and the class diagrams). The proposed method consists of the four tasks. At first, "development of the UML system specification" task describes the information related to the system's element, configuration, and control. At second, "development of the hazard scenario using STPA" task decides the accidents, hazards, SCs, and hazard scenarios related to the target system. At third, "development of the sequence diagrams corresponding to the hazard scenario and the assignment of the HCF to the classes" task develops the sequence diagrams corresponding to the hazard scenario based on the information of the use-case diagrams and the class diagrams of the ECSW. As a result, the portions that are the causes of the hazards (HCF) are clarified. At last, "conduction of the FMEA for each HCF" task conducts FMEA to each HCF, evaluates the negative impacts of the accident, and conducts the countermeasures that do not occur the HCF (not to occur the hazards), if necessary.

Tasks that consist of the proposed method

Development of the UML system specifications: "Development of the UML system specifications" task develops the use-case diagrams and the class diagrams for the target system. Here in after, those diagrams are called the UML system specifications. Use-case diagrams describe the target ECSW and the apparatuses (hardware) that have the interactions between the ECSW. The apparatuses are used when developing the sequence diagrams in "development of the sequence diagrams corresponding to the hazard scenario and assignment of the HCF to the classes ". The class diagrams describe the classes and the methods in ECSW. Those are used when developing the sequence diagram similarly.

Development of the hazard scenario: "Development of the hazard scenario using STPA" task decides the HCFs considering the UML system specifications, accidents, hazards, and HCs and develop the hazard scenarios.

At first, the target accident is decided considering the usage of the target system. The hazard that causes the accident and the conditions that the hazard causes the accident are decided. Then the SCs are defined based on the conditions that hazard causes the accident.

At second, the CSD is developed from the use-case diagrams and the class diagrams in the UML system specifications. The components in the CSD are the actors in the use-case diagrams and the classes in the class diagrams. The CAs between the components shows the method invocation between the classes that have the relations, and the direction of the CA corresponds to the direction of the inductivity. The data between components shows the return value of the invocated method. Figure 5 shown the correspondence between the UML system specifications and CSD.

At third, UCAs are derived from all combinations of CAs in the CSD and "the 4 keywords to identify the UCAs that cause the hazards". Table 1 shows the diagnostic table that is used

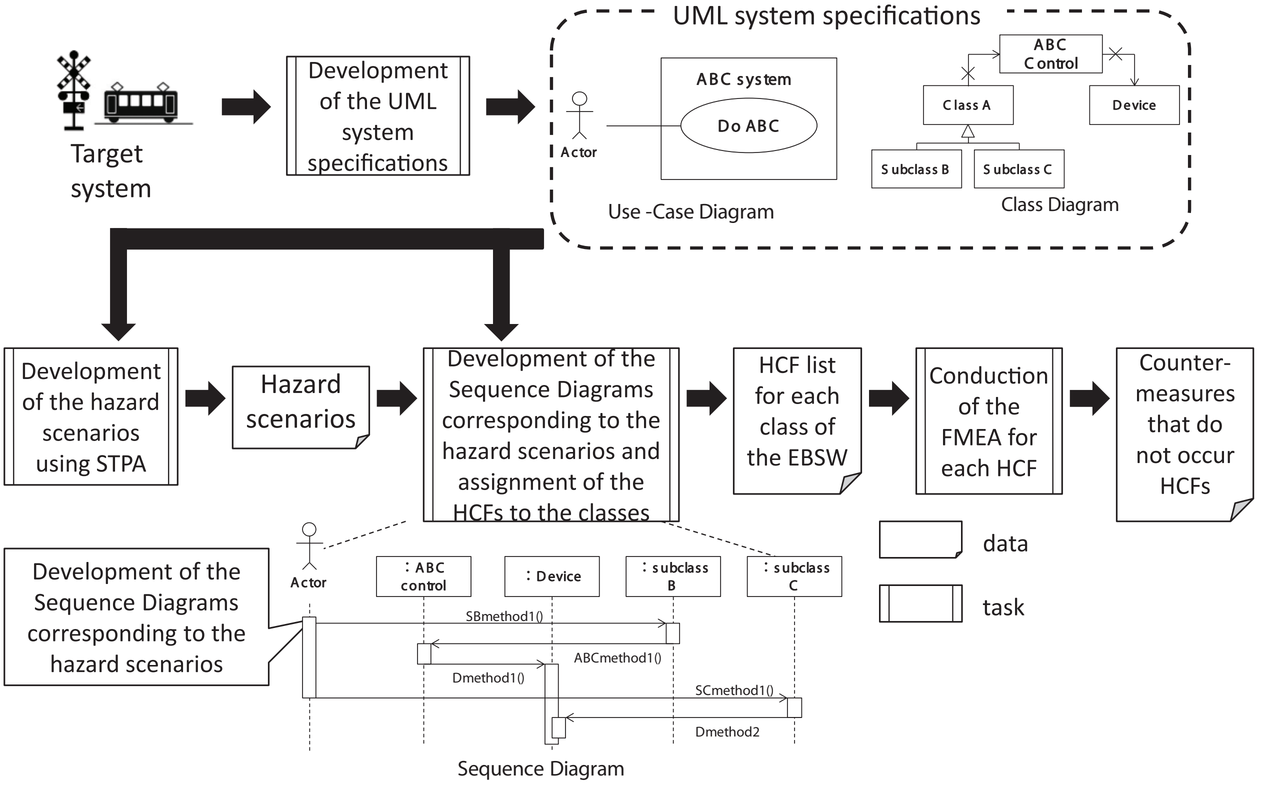
Target system
Development of the UML system specifications
UML system specifications
ABC system
Do ABC
Actor
Use -Case Diagram
ABC Control
Class A
Device
Subclass B
Subclass C
Class Diagram
Development of the hazard scenarios using STPA
Hazard scenarios
Development of the Sequence Diagrams corresponding to the hazard scenarios and assignment of the HCFs to the classes
HCF list for each class of the EBSW
Conduction of the FMEA for each HCF
Counter-measures that do not occur HCFs
data
task
Development of the Sequence Diagrams corresponding to the hazard scenarios
Actor
: ABC control
: Device
: subclass B
: subclass C
SBmethod1()
ABCmethod1()
Dmethod1()
SCmethod1()
Dmethod2
Sequence Diagram

Figure 4: Outline of the proposed method.

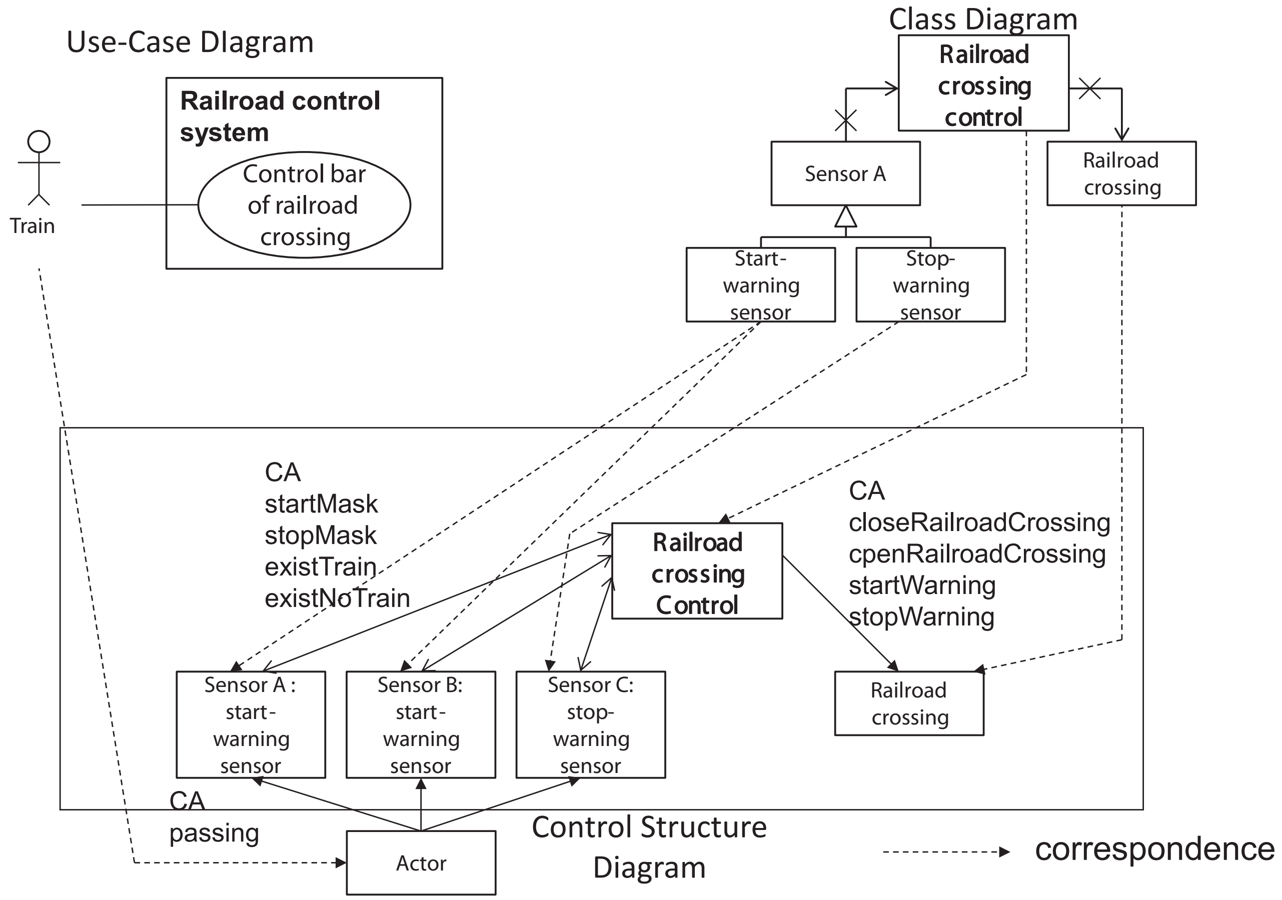

Figure 5: Correspondences between UML system specifications and CSD.

Table 1: Diagnostic table that is used for identifying UCAs ([14], P8, Table 2.4-1 modified).

	Control action	Not providing causes hazard	Providing causes hazard	Too Early/too late wrong order causes hazard	Stopping too soon/ applying too long causes hazard
1	Control action	(Condition)	(Condition)	(Condition)	(Condition)
	……	…..	…..	…..	…..

for identifying UCAs. The SCs that conflict with the UCA are written in the cells of the table.

At fourth, the control loop that causes the hazards with the UCA and CSD is identified, 11 guide words that have the possibility to become HCF are applied to UCA in the control loop, and the combination of the UCA and the guide word are evaluated whether it would be a hazard. In the case that it becomes the hazard, the conditions (HCFs) are investigated and clarified. Furthermore, the process leading to the hazard is defined as the hazard scenario.

Development of the sequence diagrams corresponding to the hazard scenario and the assignment of the HCF to the classes

"Development of the sequence diagrams corresponding to the hazard scenario and assignment of the HCF to the classes" task develops the sequence diagrams corresponding to the hazard scenario. The lifelines in the sequence diagrams are the actors in the use-case diagrams and the classes in the ECSW. The messages sent and received between the lifelines are the method of the class in the ECSW. The direction of the messages corresponds to the direction of the inductivity in the class diagrams. As a result of sending and receiving the messages according to the developed sequence diagrams, the hazard occurs. Therefore, the execution and/or non-execution of the method according to the sequence diagram are considered as the HCFs, and those HCFs are assigned to the methods in the class that receives the message. As assigning HCFs into methods in the classes for all hazard scenarios, the HCFs (methods) in each class are clarified. Figure 6 shows an example of assigning HCFs into the classes.

Conduction of the FMEA for each HCF

"Conduction of the FMEA for each HCF" task conducts functional level FMEA to the HCFs that are assigned to the methods of each class and evaluates the negative impact to the ECSW when HCF occurs. In the case that the negative impact is big, the causes of the HCFs are clarified and the countermeasures that reduce the negative impact are planned and conducted.

The function level FMEA for the ECSW is explained [9]. The failure modes of the ECSW are that the methods of the ECSW do not perform the original functionalities. Because the ECSW is software, there is no case that the ECSW does not perform the functionalities by aging (deviation of the function). The reasons why the ECSW does not perform the functionalities are the case that the function is used incorrectly (deviation of

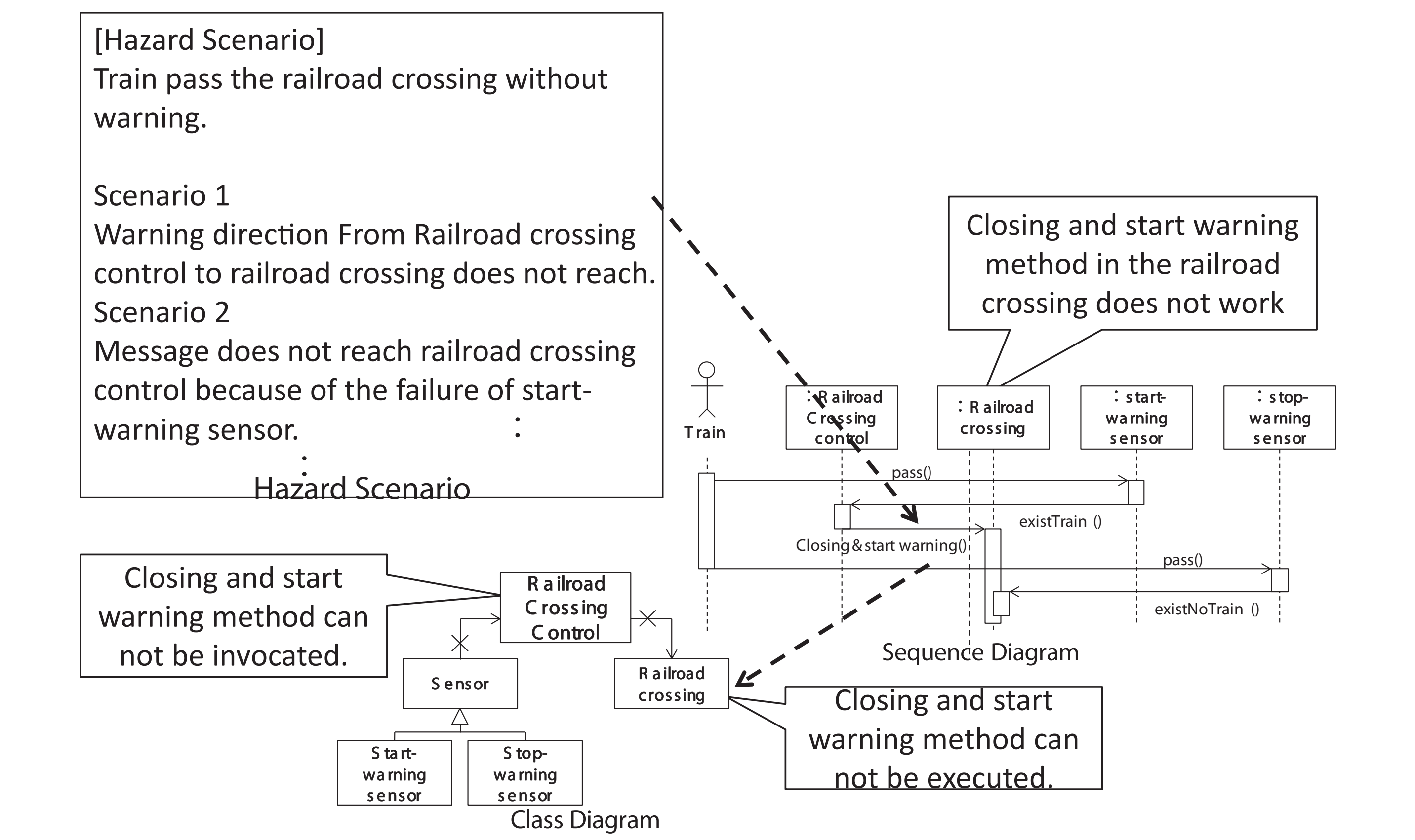
[Hazard Scenario]
Train pass the railroad crossing without warning.
Scenario 1
Warning direction From Railroad crossing control to railroad crossing does not reach.
Scenario 2
Message does not reach railroad crossing control because of the failure of start-warning sensor.
:
:
Hazard Scenario
Closing and start warning method in the railroad crossing does not work
Train
: Railroad Crossing control
: Railroad crossing
: start-warning sensor
: stop-warning sensor
pass()
existTrain ()
Closing & start warning()
pass()
existNoTrain ()
Sequence Diagram
Closing and start warning method can not be invocated.
Railroad Crossing Control
Sensor
Railroad crossing
Start-warning sensor
Stop-warning sensor
Closing and start warning method can not be executed.
Class Diagram

Figure 6: Example of assigning HCFs into the classes.

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

the execution conditions) and/or that the data outside of the range are inputted (deviation of the use conditions). Those are considered as the failure modes of the ECSW. So, the standard failure modes and standard safety countermeasures are decided by analyzing the FMEA results for the existing systems. Table 2 shows the list of the standard failure modes and standard safety countermeasures.

FMEA procedure for the ECSW is as follows. The method that is HCF in the class is investigated whether each standard failure mode can be applied. In the case that applicable standard failure modes exist, the standard safety countermeasures corresponding to the standard failure mode are selected, and the countermeasures are applied to the methods. Finally, the severity, the incidence, and the discovery rate of the method are decided. In the case that the degree of risk priority can be acceptable, selection and application of the safety countermeasures are finished. The risk evaluation matrix shown in Figure 7 is used to decide the risk priority.

Application and evaluation of the proposed method

The safety analysis for the railroad crossing control system is conducted to evaluate the proposed method. The subsection A describes the outline of the application case, and the subsection B describes the application results and the evaluation.

Outline of the application

The safety analysis for the railroad crossing control system is conducted. The railroad crossing control system is as same as the system that the Information-technology Promotion Agency (IPA) uses as an analytical example for conducting STPA [15]. Because the IPA example does not describe the ECSW that controls opening/closing the railroad crossing and rumbling/stopping the alarm device, the authors assume the configuration of the ECSW. Figure 8 shows the outline of the railroad crossing. The railroad crossing consists of the control apparatus, the railroad crossing & the alarm device, and the sensors (two alarm start sensors, such as A and B, and one alarm stop sensor, such as C. Those sensors cannot detect the direction of the train.). The requirements for the railroad crossing control system are as follows.

- When the ECSW detects the train using the alarm start sensors A or B, the ECSW starts alarm after a certain period of time.
- When the ECSW detects the train using alarm stop sensor C, the ECSW stops the alarm after a certain period of time.
- When the train moves from A to C, the alarm start sensor B is masked (not to detect the train).
- When the train moves from B to C, the alarm start sensor A is masked (not to detect the train).

Figure 9 shows the outline of the railroad crossing control

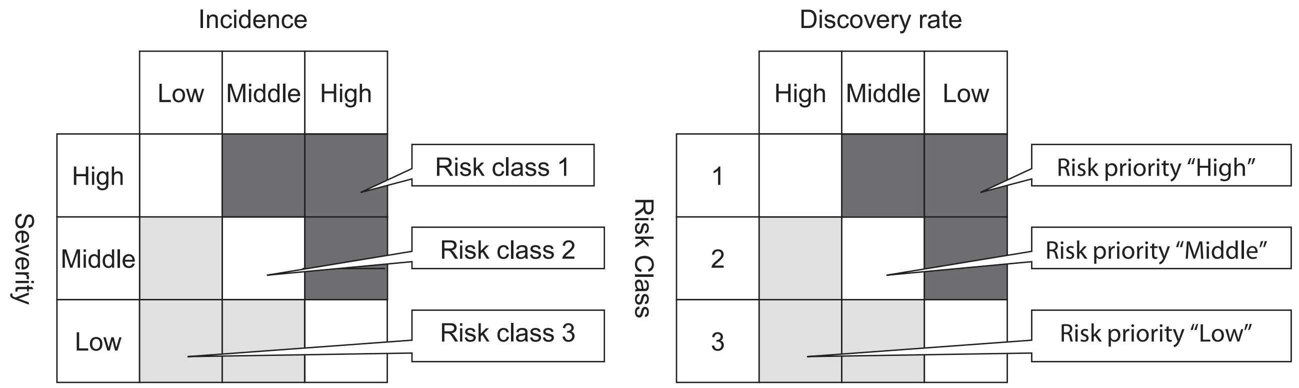

Figure 7: Risk evaluation matrix ([5], P.121, Fig.M3.5).

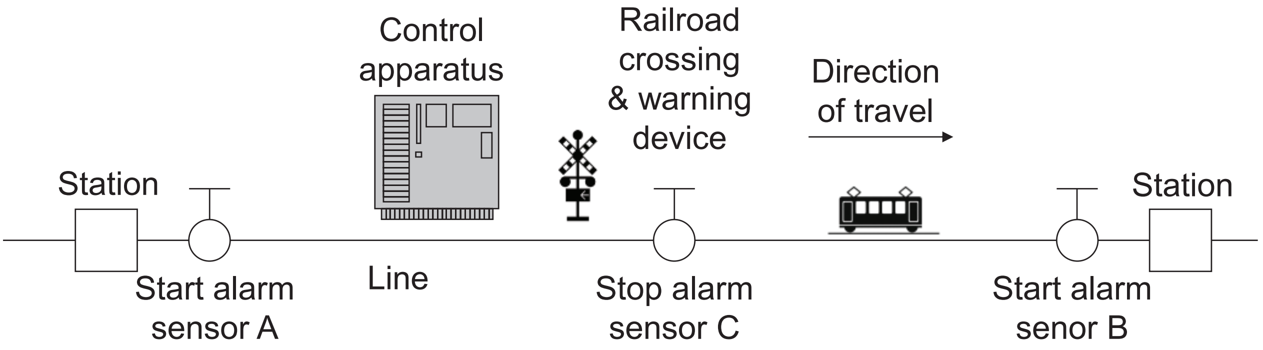

Figure 8: Outline of the railroad crossing ([14], P.11, Fig3.3-1(modified)).

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

Table 2: Standard failure modes and safety countermeasures.

Group	Standard Failure Mode	Failure Example	Countemeasure Policy	Standard Safety Countemeasures
Start up	The startup conditiond 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, set the conditiona as to
			Conduct multiple checks when startup	
			Conduct the startup check	Display the startup status
Termination	The termination conditions for functions are not prepared	Related operations cannot be conducted, an improper system status exists	Review the termination conditions for functions	Add the confirmation procedure for termination conditions to the SOP, set the conditions whether or not to terminate
			Conduct multiple checks upon termination	
			Conduct termination check	Display the termination status
			Transit to the safe status for top priority	Add the emergency stop function
Input/Output	Instructions on SOP misread	Improper products are manufactured, an improper system status exists	Conduct multiple checks on SOP	Conduct double checks on SOP
			Improve the visibility of SOP indications	Integrate the SOP format
	Indications on HMI misread	Improper products are manufactured, an improper system status exists	Conduct multiple checks upon HMI	Conduct double checks on HMI
			Improve the visibility of HMI	Integrate the HMI format
			Check the content of HMI	Add the reconfirmation function
	Mistake in checking products	Improper products are manufactured	Conduct multiple checks on products	Conduct double checks on products
	Past data is lost	Data related to quality is lost	Notify when data is lost	Add the Warning function for past data loss
	Latest data is lost	Data related to quality is lost	Notify if there is a data loss risk	Add the Warning function of the latest data loss
	An inputting error	Improper consignments are manufactured, an improper	Multiple checks on input data	Conduct double checks on setting data
Calibration	Long time intervals for fucntion calibration	A wrong measurement is done improper prodcuts are	Conduct periodic reviews	Shorten time intervals for functioncalibration
Qualification	Wrong operation authority	Proper opertaions cannot be done, improper products are manufactured	Confirm the qualification before operation	Confirm operation authority before operation
			Do not set improper authority	Review authority periodically
Backup	Memory device problem	Data disappears, data realted to quality is lost	Multiplex data save	Multiplex memory devices
			Shorten backup intervals	Conduct backup operations periodically
	Insufficient backup	Data disappears, data realted to quality is lost	Conduct proper backup operations	Organize the backup procedure in the SOP
			Shorten backup intervals	Shorten backup time intervals
Program	Unexpected amount of data is acepted	Data can not be updated	Reliize faster processing	Reliize faster updatingprocessing
		Data can not be updated	Develop faster devices	Install faster memory devices
	The upper limit of calculation precision is confirmed	Improper products are manufactured	Increased significant digits	Utilize double-precision variables
	The lower limit of calculation precision is confirmed	Improper products are manufactured	Increased significant digits	
	Divided by zero	Operation is suspended	Give a warming Division of zero	Add the warning function for a small divisor
	Unexpected amount of data is accepted	Abnormal program shutdown	Refuse data	Add the restriction function for available data
			Do not input data	Add the number of available data to the SOP
	Unexpected interruption tasks occur	Abnormal program shutdown	Restrict interruption tasks	Restrict interruption tasks
			Prohibit interruption tasks	Add the restriction function for interruption tasks to the SOP
	Unexpected CPU load occurs	Program does not reponse, a slow respons	Unexpected execution requests are not sent	Add the function of displayiong CPU usage
			Refuse unexpected execution requests	Add the restriction function for accepting execution requests under CPU overload
Malicious operations or attacks	No identifation for important data	Data is removed	Take measures so that data is not removed	Introduce DLP tools
	No access control data			Add access control for data according to each user
	Data could be written	Data is falsified	Take measures so that data is not	Add e-signature, add time stamp
	Vast amounts of data sent	Related operations cannot be conducted	Data acceptance is blocked	Disconnect from the external network
	Vast amounts of requests sent		Data is selected	Install fire walls
	Illegally accessed from the outside	System is invaded	Disconnect	Discount from the external network
			Discover illegal access	Introduce IDS
				Introduce IPS
	Data with virus attached is received	System malfunctions, improper products are manufactured	Remove computer virus	Intorduce antivirus software
			Take measures so that virus does not invade	Introduce virus protection software
				Conduct virus check USB memory devices

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

system that the authors assume. The use-case diagram shows that the train actor and the sensor actors use the control railroad crossing. The class diagram shows that the railroad crossing control system consists of the railroad crossing control class, the sensor class, and the railroad crossing & alarm device. Additionally, the sensor class has two subclasses, such as the alarm start sensor and the alarm stop sensor. The railroad crossing control class decides the CAs that are sent to the railroad crossing & alarm device based on the FBD from the sensors. The sensor classes send the FBD when the sensor classes detect the train. The railroad crossing & alarm device class controls the railroad crossing and alarm device when the railroad crossing & alarm device classes receive the CA.

The outline of the case studies is explained as follows

Case 1: The train crashes the pedestrian or the car (accident 1: A1), the railroad crossing does not close when the train exists on the railroad (hazard: H1).

The train from A passes the alarm start sensor A, passes the alarm stop sensor C, and stops. Then the train is detached into two parts, such as the front part and the rear part. The front part goes to B, and the rear part returns to A.

Case 2: Accident and hazard is as same as Case 1.

The train from A passes the alarm start sensor A, but the execution of the closeBar&start alarm method is delayed for some reason. After the train passes the alarm stop sensor C, the execution of the closeBar&startAlarm method starts tardily.

Case 3: Accident and hazard is as same as Case 1.

The first train from A passes the start alarm sensor A and passes the stop alarm sensor C. Then the start alarm sensor A and B are masked. The second train passes the start alarm sensor A immediately after the first train passes the stop alarm sensor.

Application and evaluation of the proposed method

The results of the evaluation are described below.

Case 1

At first, the UML system specifications that are shown in Figure 9 are developed. STPA is conducted by inputting the UML system specification. The following task number two to five are the same procedure in section 3.B.2).

At second, the accidents, hazards, SCs are identified. In this case, it is considered that the accident is "the train crashes the pedestrian or the car ", the hazard is "the railroad crossing does not close when the train exists on the railroad ", and the SC is "the railroad crossing must close when the train exists on the railroad (SC1)".

At third, the CSD is developed. The components of the railroad crossing control system are the railroad control, the railroad crossing & alarm device, the start alarm sensor A and B, the stop alarm sensor C, and the train. All CAs, FBD and input/output information between those components are described into the CSD. Figure 10 shows the CSD.

At fourth, the UCAs are derived. The guide words that identify the UCA are applied to the CAs in the CSD of Figure 10, and the UCAs are clarified. Table 3 shows the results of the extracts of the UCA. Here in after, the case that "The train passes the railroad crossing when not rumbling warning. (the bar of the railroad crossing does not close.) [UCA1], [SC1 violation]" is analyzed.

At fifth, it investigates whether the UCA causes the hazard (whether the UCA violates the SC). The 11 guide words that identify the HCF are applied to the UCAs in the CSD one by one, and each UCA is investigated whether it causes the hazard. Figure 11 shows the results that the guide words that identify the HCFs are assigned to the CSD. As a result, it is found that six guide words are applicable to the railroad crossing control system. Here, those six guide words are applied to all UCAs, and it is investigated that the UCAs cause the Hazards. Table 4 shows the result. As for the UCA1, UCA1 causes the hazard when applying the guide word (2) "inappropriate, inefficient or missing control" and the guide word (4) "process input missing or wrong ". Concretely, as for the guide word (2), it

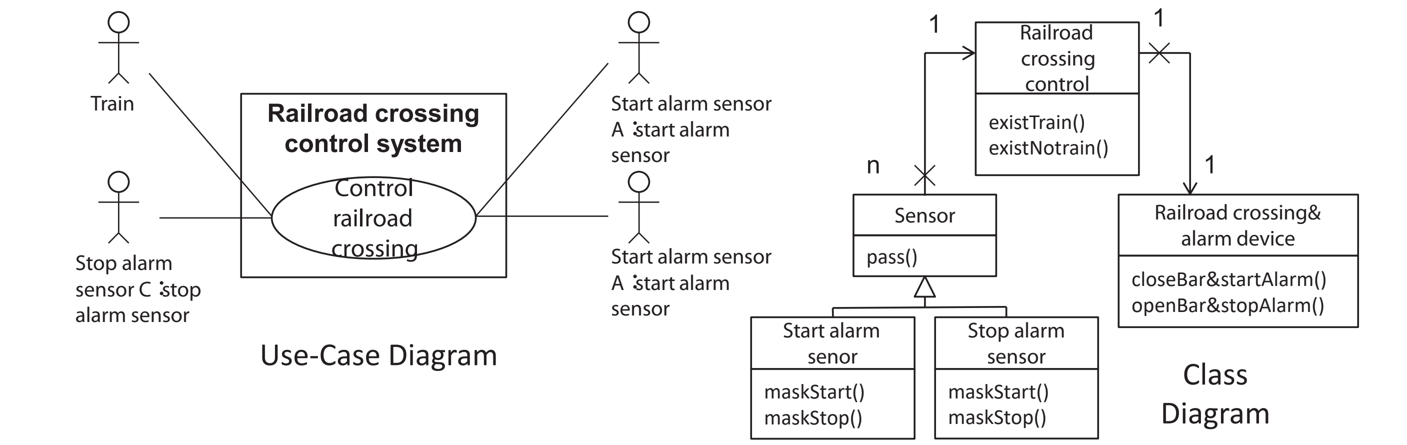
Train
Railroad crossing control system
Control railroad crossing
Start alarm sensor A :start alarm sensor
Stop alarm sensor C :stop alarm sensor
Start alarm sensor A :start alarm sensor
Use-Case Diagram
1
Railroad crossing control
existTrain()
existNotrain()
1
n
1
Sensor
pass()
Railroad crossing& alarm device
closeBar&startAlarm()
openBar&stopAlarm()
Start alarm senor
maskStart()
maskStop()
Stop alarm sensor
maskStart()
maskStop()
Class Diagram

Figure 9: Outline of the railroad crossing control system.

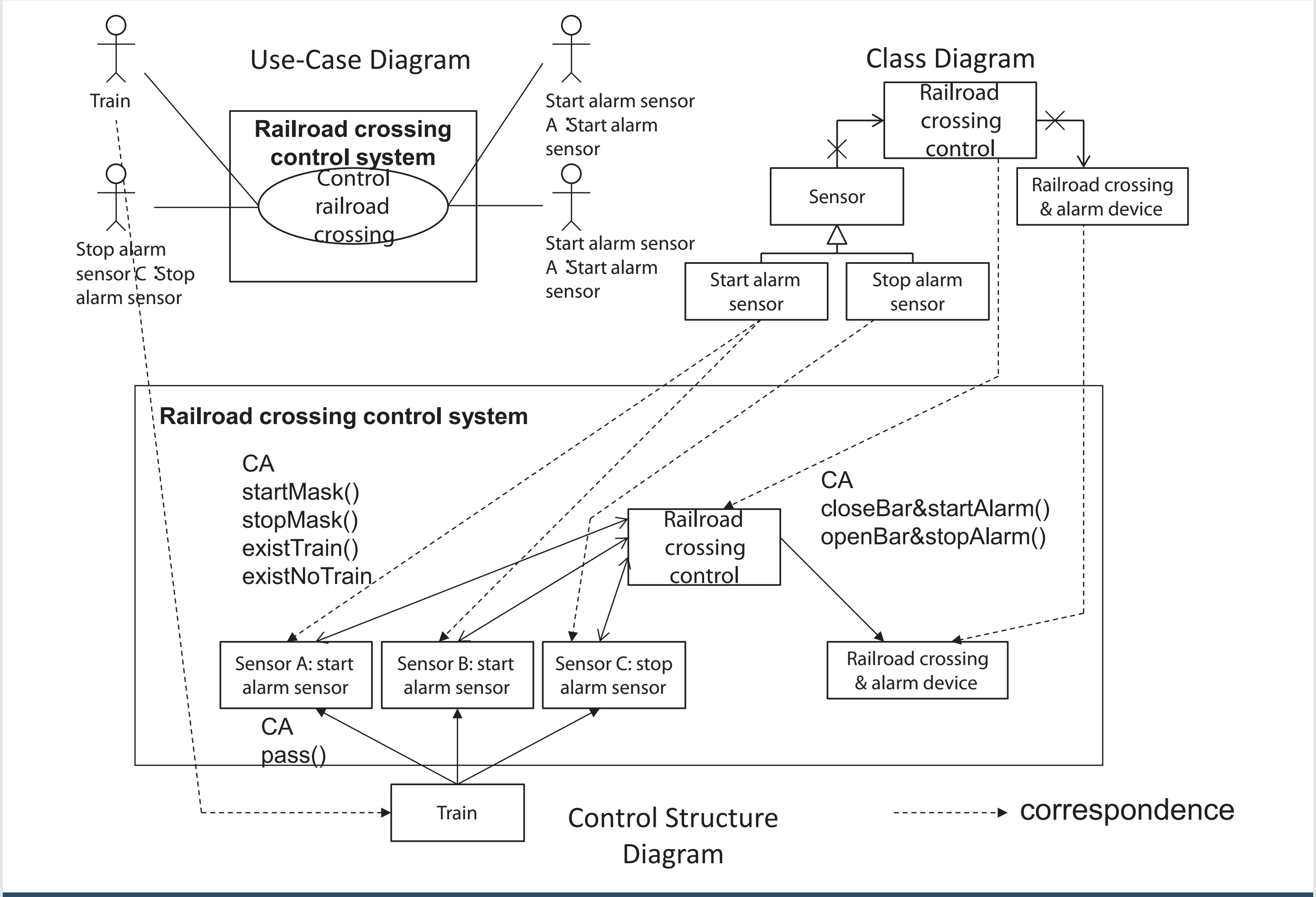
Use-Case Diagram
Train
Railroad crossing control system
Control railroad crossing
Start alarm sensor A :Start alarm sensor
Start alarm sensor A :Start alarm sensor
Stop alarm sensor C :Stop alarm sensor
Class Diagram
Railroad crossing control
Sensor
Railroad crossing & alarm device
Start alarm sensor
Stop alarm sensor
Railroad crossing control system
CA
startMask()
stopMask()
existTrain()
existNoTrain
Railroad crossing control
CA
closeBar&startAlarm()
openBar&stopAlarm()
Sensor A: start alarm sensor
Sensor B: start alarm sensor
Sensor C: stop alarm sensor
Railroad crossing & alarm device
CA
pass()
Train
Control Structure Diagram
correspondence

Figure 10: CSD of the railroad crossing control system.

Table 3: Extracted results of UCAs ([14], P.20, Table 4.4-2(modified)).

	Control Action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/appliying too long causes hazard
1	Close & start warning	(UCA1) The train passes the railroad crossing when not rumbling warning. (the bar of the railroad crossing does not close.) (SC1 violation)	The warning rumbles when the train does not come.	(UC2)The train arrives the railroad crossing before rumbling the warning. (Closing bar is too late.) (SC1 violation)	Since the startMAsk instruction is continued, the warning rumbles continously even if the stopWarning instruction is issued after passing through the train.
2	Open & stop warning method	After the train passes the railroad crossing, the warning rumbles.	(UCA3) The warning stops rumbling when the startMask instruction is invocated. (SC2 violation)	(UCA3) the warmin stops before the train passes the railroad crossing. (it is too early to open the bar of the railroad crossing after closing the bar.) (SC2 violation)	(UCA1)Since the stopWarning instruction continues after the train passes the sensor, the warning does not rumble even if the next train accesses. (competing with thw stop and start instruction) (SC1 violation)
3	startMask method (Mask enable)	When the train that passes A and c arrives B, the warning rumbles again.	(UCA4) When the train does not arrive, thw startMask instruction invocates and the warning does not rumbling. (SC1 violation)	(UCA5)When the maskStart instruction to the stop-warning sensor is delayed and is not issued before the train passes the sensor, the maskStart instruction will remain and the warning will not be rumbling in the case that two trains in the opposite direction access continuously. (SC1 violation)	(UCA6)The maskStart instruction continues to be issued after the train passes the start-Warning sensor in the opposite side, and the warning does not rumble even if the opposite train acceses. (SC1 violation)
4	stopMask method (Mask disable)	(UCA6)As the stop maskStart instruction is not issued to the start sensor on the opposite side, the sensor does not start the warning even when the opposite train accesses (including the case that the train turn back after issuing the maskStart) (SC1 violation)	The warning rumbles again.	Issuing the stopMask imstruction start rumbling again, before the train passes B.	When the maskStop instruction completes with maskStart instruction, the warning rumbles again because of the non-mask operation.

is considered the following situations; “the control action for the railroad crossing control when the train turns after passing the railroad is inappropriate and it causes the hazard” or “the competition between the continuing to stop the alarm and the indicating to start the alarm causes the hazard”. As for the guide word (4), it is considered the following situation; “As a result of the defect of the start alarm sensor A, the loss of the message from start alarm sensor A to the control apparatus causes the hazard”. The hazard scenarios corresponding to those cases are developed. Figure 12 shows the hazard scenario in the case that “the control algorithm for the railroad crossing control when the train turns after passing the railroad is inappropriate and it causes the hazard”.

At sixth, the sequence diagrams corresponding to the

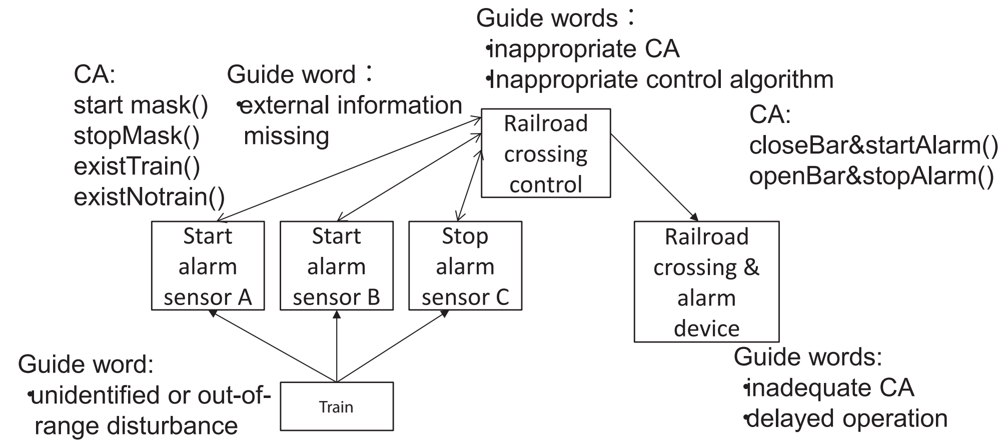

Figure 11: Results if assigned guide words to CSD.

Table 4: Hazard scenarios derived from UCA and guide words ([14], P23, Table4.5-2(modified)).

	1. Control input or external information wrong or missing	2. Inapproprite, ineffective or missing control action	3. Delayed operation	4. process input missing or wrong	5. Unidentified or out-of-range disturbance	6. Inadequate Control Algorithm
(UCA1)The train passes the railroad crossing when not rumbling warning. (the bar of the railroad crossing does not close.)		.Inappropriate control for the train that truns after the railroad crossing passes. .Competition with the continuation of stop warning and the new issue start warning.		.The failure of the sensor A causes the missing of the instruction from A to the railroad crossing control.		
(UCA2)The train arrive the railroad crossing before rumbling warning. (closing the bar is too late.)			.Delay of the warning device.			.Delay of the action of the control apparatus.
(UCA3) Rumbling warning is stopped before the train passed. (Opening the bar is too early after closing the bar.)					C causes the short circuit by the distrubance before the train arrives the railroad crossing after train passes A.	
(UCA4) When the train does not arrive, the startMask instruction invocates and the warning does not rumbling.		.Inappropriate state control of the railroad crossing.				.Inappropriate state control of the railroad crossing.
(UCA5) The warning does not rumble when the train comes because of the delay of issuing the markStop instruction.			.Delay of issuing the mask stop instruction with the no support for the high speed train.	.Distrubance by the obstacle on the rail.		.Inappropriate operation of the control apparatus causes the delay ini issuing the maskstop instruction.
(UCA6)The maskStart instruction issue too late.	.Inappropriate external input (disturbance) causes missing of stopMask instruction.	. The delay of issuing the instruction for the control appratus causes missing the stopMask instruction	.Inappropriate state control causes the missing maskStop instruction.	.Inappropriate external input causes the missing of maskStop instruction.		.Delay of the correspondence for non-steady-state operation causes the missing maskStop instruction.

hazard scenario are developed, and the HCFs are assigned to the classes. This task is the same as the task stated in the section3.B.3) Here, the sequence diagram when after the train from A turns to A after the train passes the stop-warning sensor C is developed. Figure 13 shows the details of the hazard scenario. Figure 14 shows the sequence diagrams when it occurs. In Figure 14, after the train passes the stop alarm sensor C, the start alarm sensor A and B are masked, the rear part of the train turns and passes the start alarm sensor A. At this time, as the bar of the railroad crossing is open and the alarm device stop rumbling, even if the railroad crossing control issues the new CA of openBar&stopAlarm method to the railroad crossing apparatus, the railroad crossing does not work. Consequently, because the train enters the railroad crossing when the bar of the railroad crossing is opened, it becomes the hazard. Here, it is assumed that the functions of the railroad crossing & alarm device class, the start alarm sensor class, and the stop alarm sensor class are simply sent CAs to the apparatuses through the input/output interface and there is no trouble of the hardware. As a result, the events of the hazard scenario are assigned only to the railroad crossing control class (HCFs are assigned to the methods in the railroad crossing control class). Considering the sequence diagram, the methods that are assigned to this class are existTrain method and existNoTrain method.

At seventh, the FMEA is conducted considering the sequence diagrams. The existTrain method invokes a closeBar&startAlarm method of the railroad crossing & alarm device class. Even if this method is invoked, the bar of the railroad crossing is still closed, and the alarm device is only rumbling. Therefore, as there is a low possibility when this hazard occurs, the countermeasures for this event are not applied. On the other

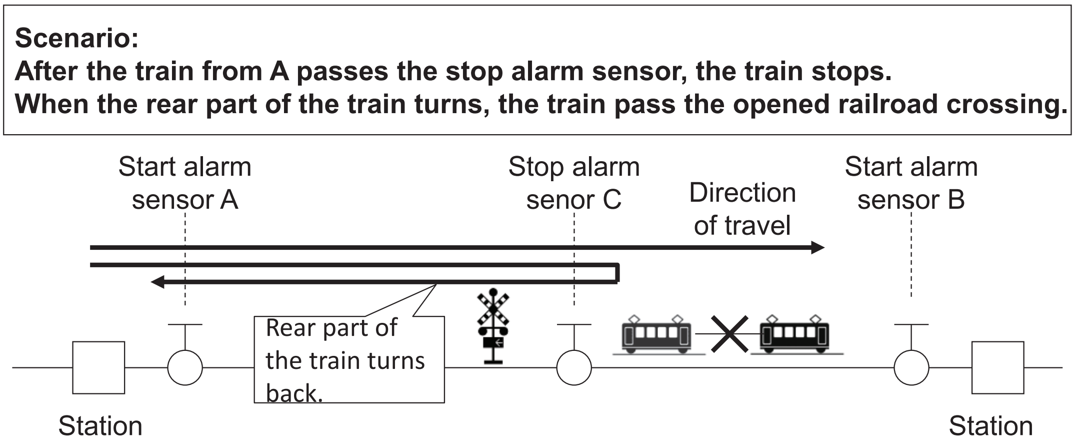
Scenario:
After the train from A passes the stop alarm sensor, the train stops.
When the rear part of the train turns, the train pass the opened railroad crossing.
Start alarm sensor A
Stop alarm senor C
Direction of travel
Start alarm sensor B
Rear part of the train turns back.
Station
Station

Figure 12: Example of hazard scenario ([14], P.24, Fig.4.5-2(modified)).

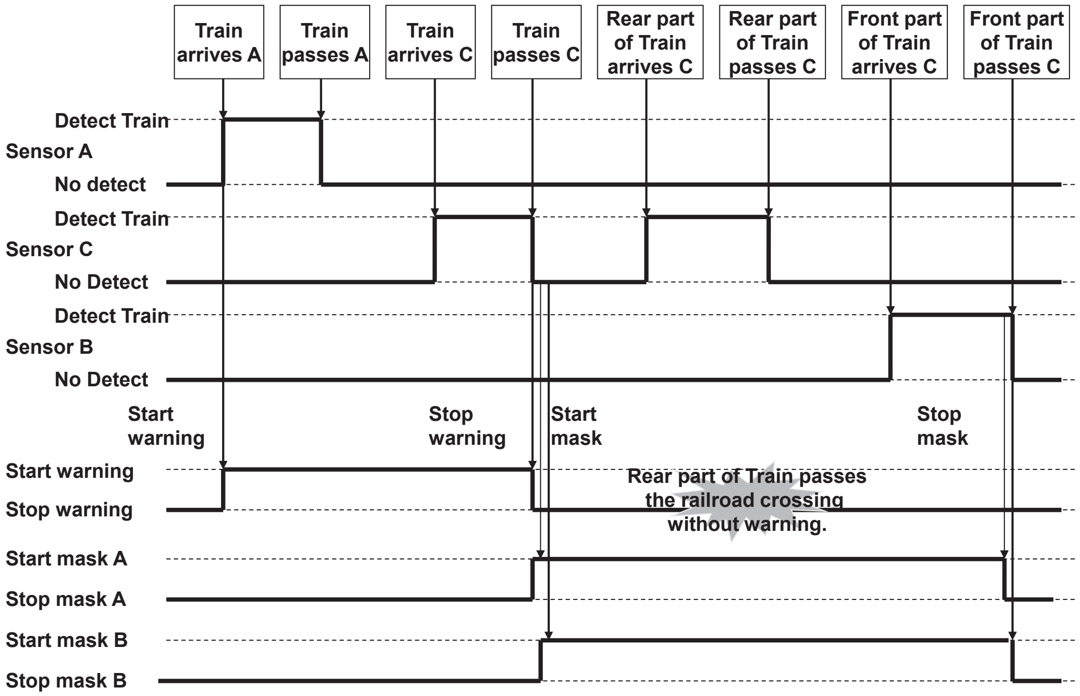

Figure 13: Details of the hazard scenario of case 1.

hand, the existNoTrain method invokes an openBar&stopAlarm method in the railroad crossing & alarm device. Generally, the existTrain method and existNoTrain method should be carried out in pairs. Additionally, the existTrain method and existNoTrain method should be invoked alternately. "The startup conditions for functions are not prepared" in Table 2 can be applied. Therefore, the setting of the startup conditions and the setting of non-startup conditions are applicable as the standard countermeasures. For example, the state transition diagram is added to the railroad crossing control class (Figure 15). In the case that the message of existNoTrain method is received when the state is in the waiting the train passing, the following countermeasures are conducted; issue the emergency message to the safety supervisor (the method that issues the alarm is added to the railroad crossing control class), close the bar of the railroad crossing, and rumble the alarm. Those countermeasures reduce the rate of the incident that causes the hazard.

Case 2

The UML system specifications, the accidents, the hazards, SCs and CSD are same as the CASE 1. CASE2 corresponds to the case that "The train arrive the railroad crossing before rumbling the warning. (closing the bar is too late.) [UCA2], [SC1 violation])" in Table 3. It is investigated whether the UCA2 is the hazard or not (UCA2 violates SC1).

As a result of applying 11 guide words that identify the HCFs to CAs in the CSD, it is found that the case that guideword "(3) delayed operation" occurs becomes the hazard. Figure 16 shows the details of the hazard scenario, and Figure 17 shows the sequence diagrams.

In Figure 17, after the train passes the start alarm sensor A, a closeBar&startAlarm method is invocated. Because the invocation of the closeBar&startAlarm method is delayed for some reason, the train enters the railroad crossing when the

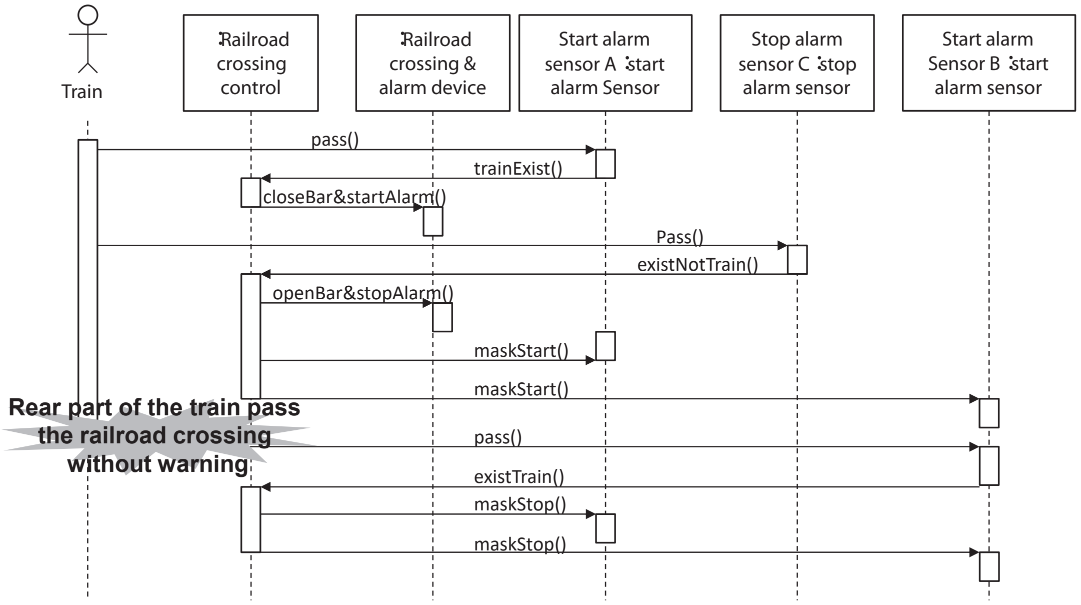
Train
Railroad crossing control
Railroad crossing & alarm device
Start alarm sensor A :start alarm Sensor
Stop alarm sensor C :stop alarm sensor
Start alarm Sensor B :start alarm sensor
pass()
trainExist()
closeBar&startAlarm()
Pass()
existNotTrain()
openBar&stopAlarm()
maskStart()
maskStart()
Rear part of the train pass the railroad crossing without warning
pass()
existTrain()
maskStop()
maskStop()

Figure 14: Sequence diagram of case 1.

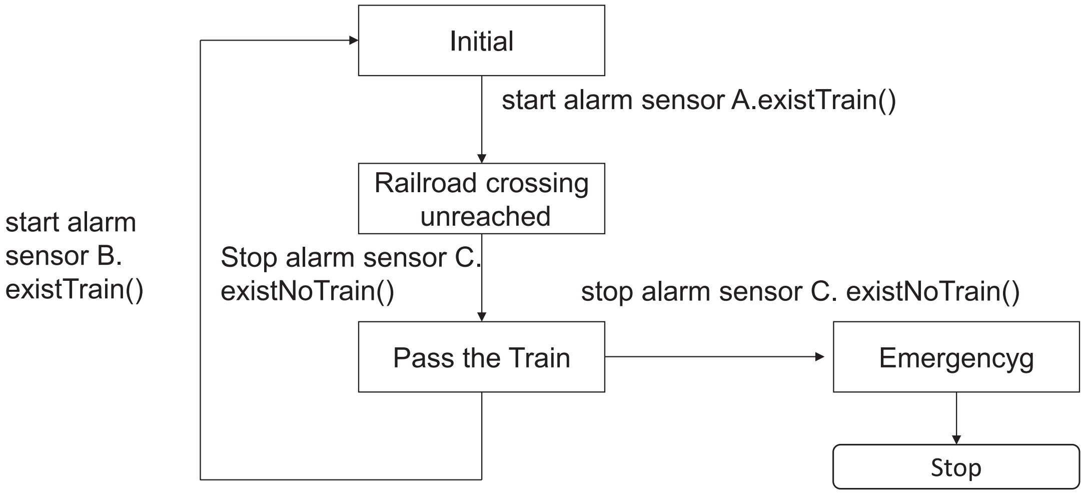
Initial
start alarm sensor A.existTrain()
Railroad crossing unreached
start alarm sensor B. existTrain()
Stop alarm sensor C. existNoTrain()
stop alarm sensor C. existNoTrain()
Pass the Train
Emergencyg
Stop

Figure 15: Added state transition diagram to railroad crossing control system.

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

bar of the railroad crossing is opened, and the alarm is stopped. The hazard occurs when the existTrain method in the railroad crossing class does not invocate the openBar&stopAlarm method in the railroad crossing and alarm device class. FMEA is conducted for the case. When the existTrain method invocates the closeBar&startAlarm method, the method must be invocated at top priority. "The startup conditions for functions are not prepared" in Table 2 can be applied. Therefore, the setting of the startup conditions and the setting of non-startup conditions are applicable as the standard countermeasures. For example, the closeBar&startAlarm method is invocated at the beginning of the existTrain method, or other methods are not invocated when the existTrain method is running. Those countermeasures reduce the rate of the incident that causes the hazard.

Case 3

The UML system specifications, the accidents, the hazards, SCs and CSD are same as the CASE 1. CASE3 corresponds to the case that "When the train does not arrive, the startMask instruction invocates and the warning does not rumbling. [UCA4], [SC1 violation]" in Table 3. It is investigated whether

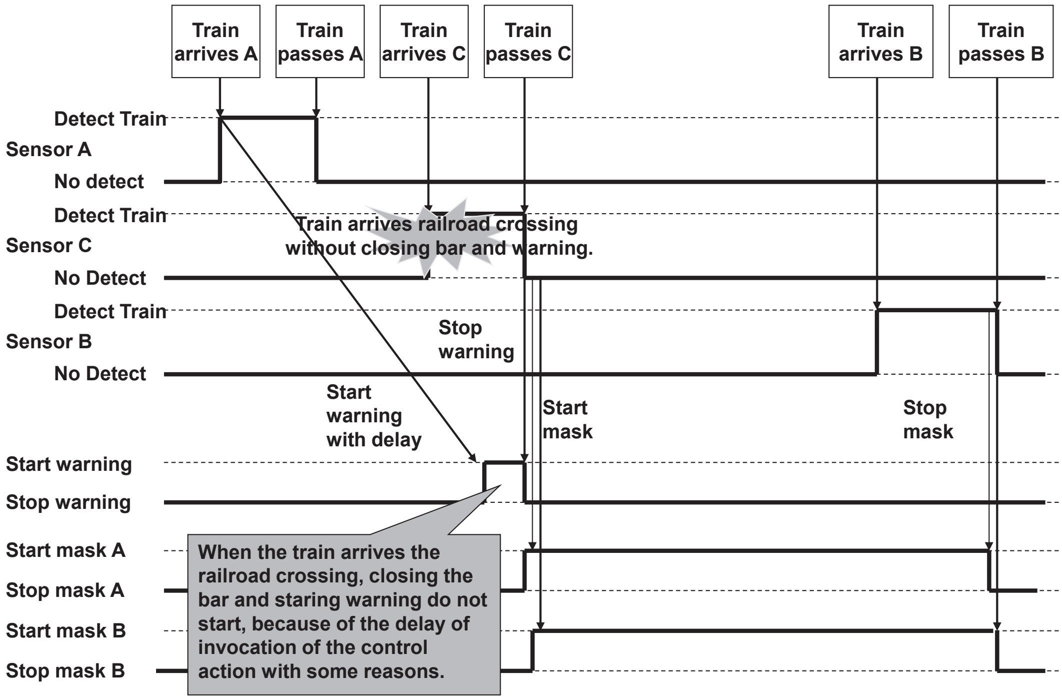

Figure 16: Details of the hazard scenario of case 2.

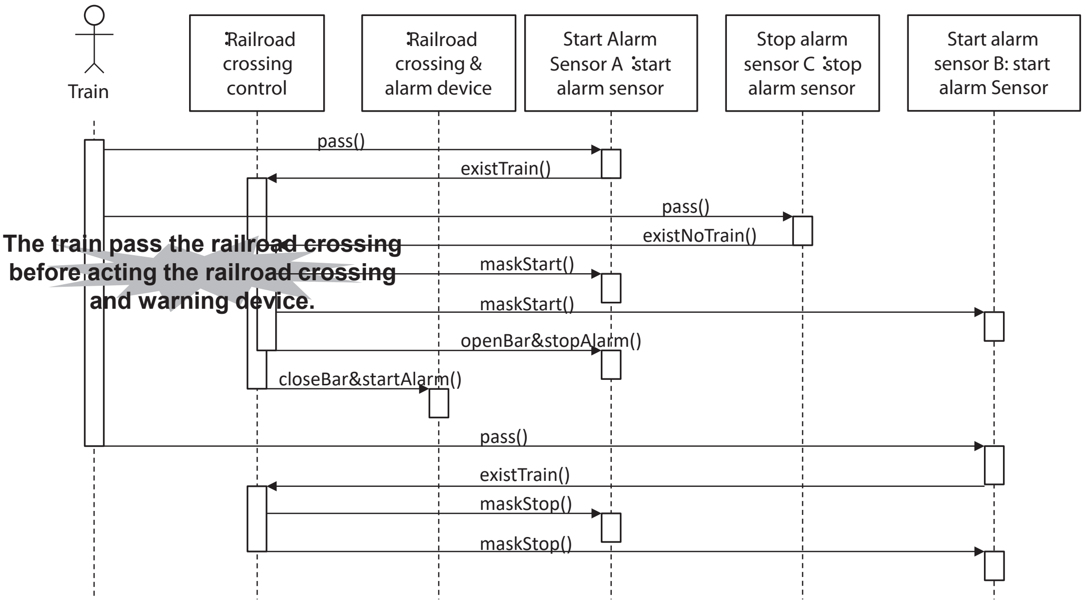
Train
Railroad crossing control
Railroad crossing & alarm device
Start Alarm Sensor A :start alarm sensor
Stop alarm sensor C :stop alarm sensor
Start alarm sensor B: start alarm Sensor
pass()
existTrain()
pass()
existNoTrain()
maskStart()
maskStart()
openBar&stopAlarm()
closeBar&startAlarm()
pass()
existTrain()
maskStop()
maskStop()
The train pass the railroad crossing before acting the railroad crossing and warning device.

Figure 17: Sequence diagram of case 2.

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

the UCA4 is the hazard or not (UCA4 violates SC1). As a result of applying 11 guide words that identify the HCFs to CAs in the CSD, it is found that the case that guideword "(2) inappropriate, inefficient or missing control action " occurs becomes the hazard. Figure 18 shows the details of the hazard scenario, and Figure 19 shows the sequence diagrams.

In Figure 19, after the first train passes the stop alarm sensor C, the start alarm sensor A and B are masked. When the first train passes the start alarm sensor B, the start alarm sensor B is released to be masked. After this situation, though the second train enters and passes the start alarm sensor A, the closeBar&startAlarm method is not invocated because the start alarm sensor A is masked. As a result, the second train enters the railroad crossing that the bar is opened and the alarm is not rumbled, and this situation becomes the hazard.

According to the sequence diagram, when the first Train passes the stop alarm senor C, the start alarm sensor A and B are masked. After this situation, as the start alarm sensor A is

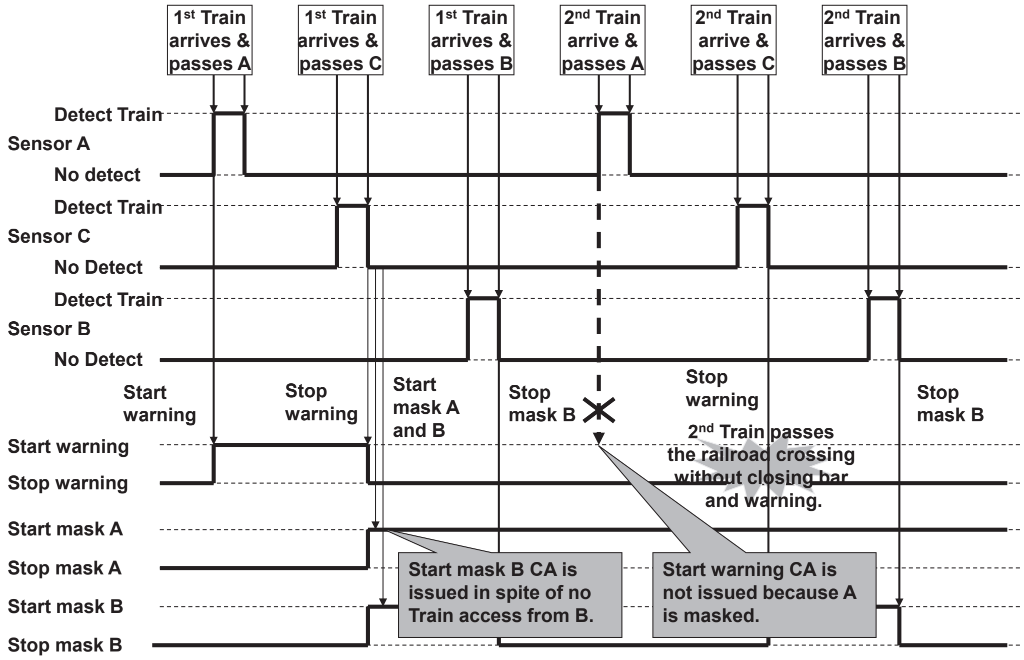
1st Train arrives & passes A
1st Train arrives & passes C
1st Train arrives & passes B
2nd Train arrive & passes A
2nd Train arrive & passes C
2nd Train arrives & passes B
Detect Train
Sensor A
No detect
Detect Train
Sensor C
No Detect
Detect Train
Sensor B
No Detect
Start warning
Stop warning
Start mask A and B
Stop mask B
Stop warning
Stop mask B
2nd Train passes the railroad crossing without closing bar and warning.
Start warning
Stop warning
Start mask A
Stop mask A
Start mask B
Stop mask B
Start mask B CA is issued in spite of no Train access from B.
Start warning CA is not issued because A is masked.

Figure 18: Details of the hazard scenario of case 3.

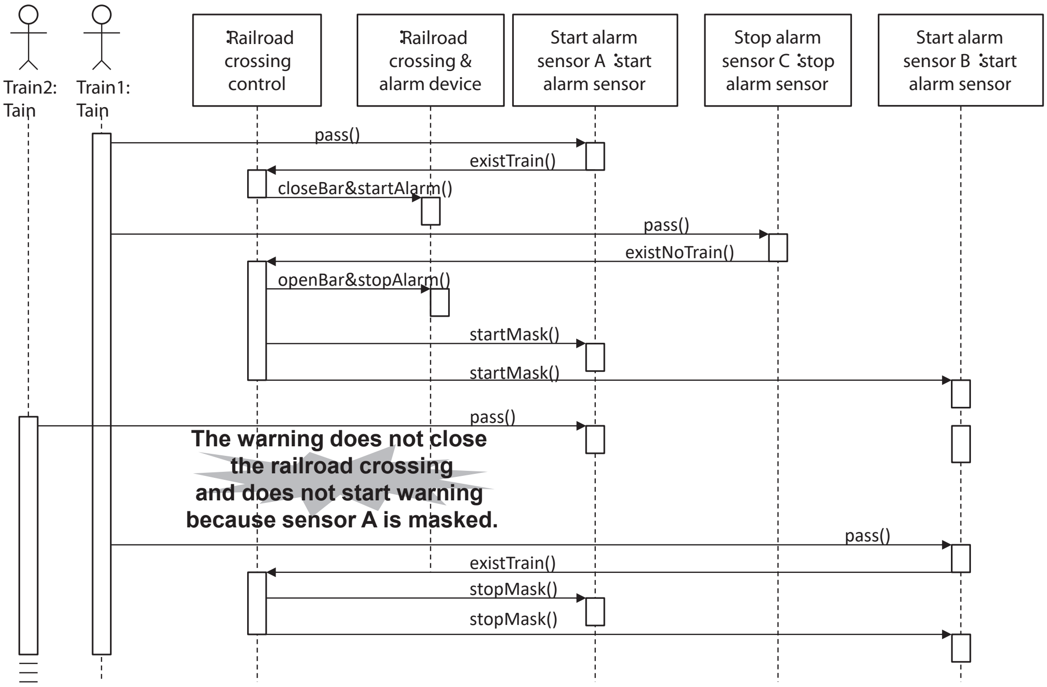
Train2: Tain
Train1: Tain
Railroad crossing control
Railroad crossing & alarm device
Start alarm sensor A :start alarm sensor
Stop alarm sensor C :stop alarm sensor
Start alarm sensor B :start alarm sensor
pass()
existTrain()
closeBar&startAlarm()
pass()
existNoTrain()
openBar&stopAlarm()
startMask()
startMask()
pass()
The warning does not close the railroad crossing and does not start warning because sensor A is masked.
pass()
existTrain()
stopMask()
stopMask()

Figure 19: Sequence diagram of case 3.

Citation: Takahashi M, Anang Y, Watanabe Y (2020) A Hazard Analysis Method for Embedded Control Software with STPA. Trends Comput Sci Inf Technol 5(1): 082-096. DOI: https://dx.doi.org/10.17352/tcsit.000029

still masked, the closeBar&startAlarm method is not invocated when the second Train passes the start alarm sensor A. This case corresponds to the failure of the execution conditions. Therefore, the setting of the startup conditions and the setting of non-startup conditions are applicable as the standard countermeasures. For example, the start alarm sensor A and B are masked and released at the same timing, and the sensors that are not involved are not masked. This countermeasure reduces the rate of the incident that causes the hazard.

It is found that the adequate countermeasures are similarly applied to the other hazard scenarios. As the result of applying the proposed method, the hazards of the railroad crossing control system can be clarified, and the appropriate countermeasures to avoid occurring the hazards can be found. Consequently, the risks that the hazards occur are reduced, and the safety of the target system becomes improved. On the other hand, because there are many hazard scenarios, it is found that an efficient method for investigating the countermeasures is required. Additionally, it is found that there is a probability that the conflicts between the countermeasures occur because the proposed method decide the countermeasures corresponding to each hazard scenario. For that reason, it is found that the method that checked the conflict between the countermeasures is required.

In these case studies, we conducted the design modifications of the ECSW to avoid occurring the hazard. Regarding this problem, it could be also possible to solve it by establishing the standard operation procedure (rules) that did not permit the detach and/or turn of the train between the sensor A and B. Actually, when deciding the countermeasures, the safety, the cost and the development time must be considered, and adequate countermeasures, such as the modification of the standard operation procedures, the design modification of hardware, or the design modification of the software, should be selected. That is, the safer mechanisms need to be developed efficiently.

Future works

This paper proposes a safety analysis method cooperating with the UML, STPA, and FMEA. The proposed method analyzes the causes of the hazards that are occurred by the interactions between the system components and proposes the countermeasures that avoid occurring the hazards. As a result of the application of the proposed method, the safer system can be developed. On the other hand, it is found that the proposed method requires a long time for analyzing hazard and planning the countermeasures. Especially, in the case when analyzing the hazards of the complex system, because the system includes many hardware and software, and the system has many hazards and the hazard scenarios, it would occur the problem that the decided countermeasures applying the proposed method have conflicts in each other. In the future, we will propose a method that describes the SCs using logical expressions and analyzes them automatically using a logical calculation. As a result, a mechanism that will be able to conduct adequate and efficient safety analysis will be developed. Additionally, we will apply the proposed method to the larger system, clarify the weak points of the method, propose the countermeasures, feedback them into the proposed method, and improve the proposed method.

Acknowledgments

This research was supported by Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science “Integrated Analysis Method for hazard caused by software interaction cooperating with multiple safety analysis methods.”

References

1. Leveson N (2011) Engineering a Safer World, The MIT Press.

2. Japanese Standards Association (2017) JIS2304 Medical Device Software - - Software Life Cycle Process, Japanese Standards Associations.

3. International Electro technical Commission (2006) ICE 62304 Medical Device Software, International Electro technical Commission. Link: https://bit.ly/2IODVVD

4. International Electrotechnical Commission (2016) ICE 82304-1 Health Software - - Part 1: General Requirements for Product Safety, International Electrotechnical Commission. Link: https://bit.ly/35LWAdv

5. International Society for Pharmaceutical Engineering (2008) GAMP5 A Risk-Based Approach to Compliant GxP Computerized Systems, International Society for Pharmaceutical Engineering.

6. International Organization for Standardization (2011) ISO26262 Road vehicles – Functional safety, International Organization for Standardization.

7. Radio Technical Commission for Aeronautics (2011) DO-178C Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics.

8. Japan Aerospace Exploration Agency (2008) JAXA JMR001 System Safety Standard, Japan Aerospace Exploration Agency.

9. Takahashi M, Nanba R, Fukue A (2012) Proposal of Operational Risk Management Method Using FMEA for Drug Manufacturing Computerized System. Transaction of the Society of Instrument and Control Engineers 48: 285-294. Link: https://bit.ly/3pJ9d0E

10. Weber W, Tondok H, Bachmayer M (2003) Enhancing Software Safety by Fault Trees: Experiences from an Application to Flight Critical SW. Proc of SAFECOMP 289-302. Link: https://bit.ly/3lL3IMI

11. Leveson N, Harvey PR (1983) Analyzing Software Safety. IEEE Transaction on Software Engineering 9: 569-579. Link: https://bit.ly/2HdVKfV

12. Leveson N, Cha S, Shineall T (1991) Safety verification of Ada Programs Using Software Fault Trees. IEEE Software 8: 48-59. Link: https://bit.ly/3lQINrK

13. Takahashi M, Nanba R (2014) A Proposal of Fault Tree Analysis for Control Programs. Proc of SICE Annual Conference 1719-1724. Link: https://bit.ly/35HIB8k

14. Pai G, Dugan J (2002) Automatic Synthesis of Dynamic Fault Tree from UML System Model. Proc of 13th International Symposium on Software Reliability Engineering. Link: https://bit.ly/3kKF4KX

15. Information-technology Promotion Agency (2016) The first step of STAMP/STPA - A New Safety Analysis Method based on the System Oriented Thinking. Information-technology Promotion Agency.