Proceedings of the SICE Annual Conference 2019
September 10-13, 2019, Hiroshima, Japan

A Proposal for a Hazard Analysis Method for Embedded Control Software Using STAMP

Masakazu Takahashi[1†], Yunarso Anang[2], and Yoshimich Watanabe0[3]

[1]Department of Computer Science and Engineering, University of Yamanashi, Kofu, Japan
(Tel : +81-55-220-{[1]8585, [3]8651}, ; E-mail: {[1]mtakahashi, [3]nabe}@yamanashi.ac.jp)
[2]Department of Computational Statistics, Institute of Statistics, Indonesia
(Tel : +62-21-819-1437; E-mail: anang@stis.ac.id)

Abstract: The software that control actions of industrial products are called embedded control software (ECSW). This research proposes a method for hazard analysis caused by ECSW using System-Theoretic Process Analysis (STPA). The outline of the proposed method is as follows. (1) The accidents and hazards that constitute the target for analysis are defined. (2) The hardware and ECSW are designed using the Unified Modeling Language (UML). (3) Unsafe control actions of the ECSW and hazard scenarios are clarified by conducting STPA using the hazard information and UML system specifications as inputs. (4) The hazard scenarios are described in detail using sequence diagrams, and interactions between the hardware and ECSW are clarified. (5) Parts of the hardware and functions of the hazard occurrence and conditions that lead to hazards are clarified by analyzing the sequence diagrams. (6) Countermeasures to prevent conditions are planned and applied to the components.

Keywords: Safety Analysis, Embedded Control Software, STPA, Hazard

1. INTRODUCTION

First, the technical terms used in this research are defined. An accident refers to an event that causes a loss for the target system. Losses imply negative effects for users, missions, or the target system. Hazard refers to a state of the system that negatively affects the target system given some negative conditions.

Recently, industrial products, such as cars and medical and aerospace apparatuses, are developed as systems that combine hardware and software (components). Though their configuration and control are complex, accidents can occur. In many cases, the hazards cause accidents that arise from the interactions between the hardware and software. This accident model is called System-Theoretic Accident Model and Process (STAMP). A safety analysis method that clarifies hazards and scenarios based on the STAMP is called STAMP-based Process Analysis (STPA) [1].

The software that control actions of industrial products are called embedded control software (ECSW). This research proposes an analysis method for hazards caused by the interactions between hardware elements and ECSW's functions using STPA. The hardware elements and ECSW's functions are called components.

The outline of the proposed method is as follows. First, the accidents and hazards are defined. Second, the hardware and ECSW configuration are described using use-case diagrams and class diagrams (UML system specifications). Next, unsafe control actions (UCAs) of the ECSW and processes leading to hazards (hazard scenarios) are clarified by conducting STPA. Hazard scenarios are then described in detail using sequence diagrams, and the interactions between the components are clarified. Moreover, components related to hazard occurrence and conditions are clarified by analyzing the sequence diagrams. Finally, countermeasures to prevent hazards are planned. These countermeasures are prepared analyzing the existing hazard countermeasures.

† Masakazu Takahashi is the presenter of this paper.

2.RELATED WORKS

2.1 Related researches

Takahashi et al. proposed a method to clarify all accidents that may occur and device countermeasures to solve them using failure mode and effects analysis (FMEA) [2]. Weber et al. proposed a fault detection method using fault tree analysis (FTA) for avionics software written in assembler [3]. Leveson et al. showed that fault tree (FT) can be developed by preparing and combining FT templates corresponding to the essential instructions of the ECSW [4, 5]. Takahashi et al. proposed rules for automatically developing FT by tracing the process that causes accidents and combining FT templates [6]. Pai et al. proposed a method that calculates the reliability of the system by inputting design specifications written in UML [7]. Though these methods can clarify the cause of failures at the component level for industrial products, the complex failures that arise from the interactions between the components cannot be dealt with.

2.2 STAMP and STPA

The STAMP model is first explained. Fig.1 shows an outline of the STAMP model. The STAMP model describes a system that consists of a 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 changes the state of the process model. The controlled process changes the inner state based on the received CA and returns the result as feedback data (FBD). If the state of the process model does not correspond to the state of the controlled process, the system is in the unsafe state, at which point hazard may occur. A diagram that describes the relationships for the target system is called control structure diagram (CSD). Unsafe CAs (UCAs) are defined by applying "the 4 keywords to identify UCAs (such as not providing, providing, too fast/too late, inappropriate execution sequence, too fast/too long)" to the CAs. Conditions under which every UCA can cause hazards are clarified. The UCA in the CSD's control loop is applied to the guide word one by one to check if it causes a hazard. Fig.2 shows "the 13 guide words used for finding a hazard in a control loop." The conditions that lead to hazards are clarified. In addition, scenarios are developed to show the processes of the hazard. Finally, countermeasures to prevent hazards are developed by considering the hazard scenarios.

3.PROPOSED HAZARD ANALYSIS METHOD

3.1 Outline of the Proposed Method

This research proposes an analysis method for hazards caused by ECSW designed based on the object-oriented design method. The hardware parts and ECSW's functions (method) are called the components. The characteristic of the proposed method is that the hazard occurrence process caused by interactions between the components are assigned to the hazard occurrence conditions (HOCs) for each component. This makes planning the hazard prevention countermeasures (HPCs) easy. By combining and applying these HPCs, the safety of the industrial product is improved.

Fig.3 shows an outline of the proposed method. The proposed method consists of six steps. Task (1) "definition of accidents and hazards" describes the target accident and hazard of the industrial product and the ECSW. Task (2) "development of UML system specification" describes ECSW use-case and class diagrams referring to the requirements of the hardware and software. These diagrams are called UML system specifications. In addition, CSDs are developed by referring to the UML system specifications. Task (3) "analysis of unsafe control actions and development of hazard scenarios using STPA" clarifies the UCAs by applying "the 4 keywords to identify UCAs that cause hazards" to CSD. In addition, by applying "the 13 guide words used for finding hazards in a control loop" to UCA, UCAs that lead to hazards are clarified. These processes are called hazard scenarios. Task (4) "development of sequence diagrams corresponding to hazard scenarios" describes sequence diagrams by referring to hazard scenarios and UML system specifications. Task (5) "clarification of the hardware and software components related to the occurrence of hazards" assigns interactions between hardware parts and ECSW's method to each component, thereby making the HOCs clear. In addition, task (6) "planning hazard prevention countermeasures for each component" plans HPCs for each component using standard HPCs.

3.2 Tasks make up the Proposed Method

3.2.1 Definition of accidents and hazards

The task "definition of accidents and hazards" defines the target accidents and hazards for analysis. The inputs for this task are the design specifications of the industrial product, ECSW requirement specifications, and operation manuals, and the output is the list of accidents and hazards.

3.2.2 Development of UML system specifications

The task "development of UML system specifications" describes UML system specifications that consists of use-case diagrams and class diagrams for the target system. The inputs of this task are the requirements of the ECSW and operation manuals, whereas the output is UML system specifications. The use-case diagrams describe the scope of the hardware and ECSW. In the proposed method, relations that signify control are described using dotted arrows from the control apparatus to the controlled apparatus, whereas relations that represent sending and receiving of data are described using solid lines from the sending apparatus to the receiving apparatus. This information is used when developing the CSD. Class diagrams describe the ECSW's classes and methods.

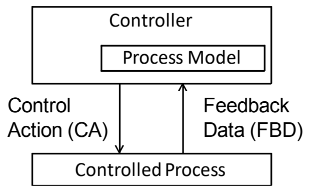

Fig.1 Outline of STAMP Model (Fig.1.1-1 in[1])

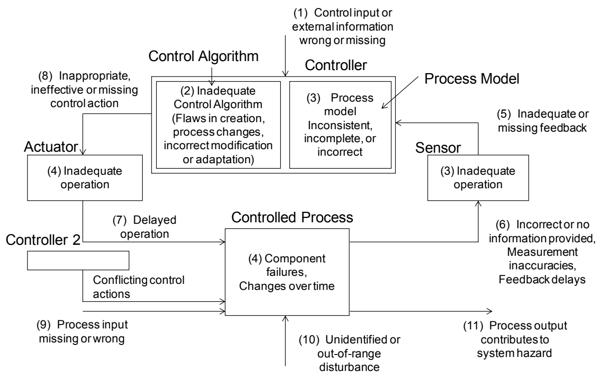
(1) Control input or external information wrong or missing
Control Algorithm
Controller
Process Model
(8) Inappropriate, ineffective or missing control action
(2) Inadequate Control Algorithm (Flaws in creation, process changes, incorrect modification or adaptation)
(3) Process model Inconsistent, incomplete, 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

Fig.2 guide words to find the causes of a hazard in a control loop （Fig.2.5-1 in [1], modified）

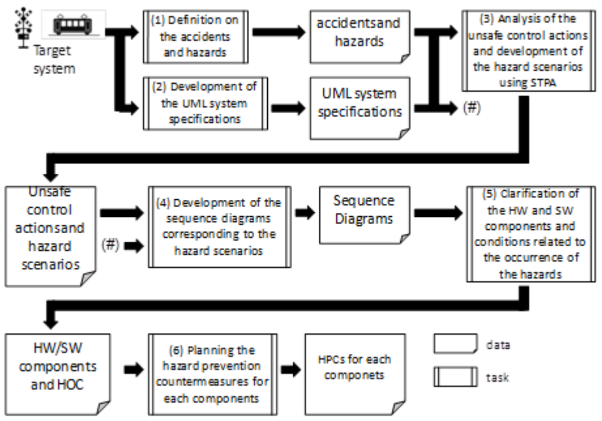
Target system
(1) Definition on the accidents and hazards
accidentsand hazards
(2) Development of the UML system specifications
UML system specifications
(3) Analysis of the unsafe control actions and development of the hazard scenarios using STPA
(#)
Unsafe control actionsand hazard scenarios
(#)
(4) Development of the sequence diagrams corresponding to the hazard scenarios
Sequence Diagrams
(5) Clarification of the HW and SW components and conditions related to the occurrence of the hazards
HW/SW components and HOC
(6) Planning the hazard prevention countermeasures for each components
HPCs for each componets
data
task

Fig.3 Outline of the Proposed Method

3.2.3 Analysis of unsafe control actions and development of hazard scenarios using STPA

The task "analysis of unsafe control actions and development of hazard scenarios using STPA" analyzes the UCAs and develops the hazard scenarios. The input to this task is UML system specifications, and the output is the CSD. First, the CSD is developed. The CSD comprises components that are actors in use-case diagrams and classes in the class diagrams. The CAs between components are method invocations between classes. The direction of the CA is the same as the direction of the relation representing the control and direction of the navigability. Sending and receiving FBD between components is a relation that implies the return value of the method. Fig.4 shows the interactions between the UML system specifications and the CSD. Second, the UCAs are derived from all combinations of the CAs in the CSD and "the 4 keywords to identify UCAs that cause hazards." The CSD serves as input for this task, whereas the output is the list of UCAs. Finally, control loops that cause hazards are identified by referring to the UCA and CSDs, "the 13 guide words used for finding hazards in control loops" are applied to the control loop, and each control loop is checked if it is a hazard or not. Processes that lead to a hazard are defined as the hazard scenarios. The HOCs and processes that lead to hazards are described in the hazard scenario. The input for this task is the UCAs, and the output is the hazard scenarios.

3.2.4 Development of sequence diagrams corresponding to the hazard scenarios

The task "development of sequence diagrams corresponding to the hazard scenarios" describes the details of the hazard scenarios using sequence diagrams. This task defines the details of the hazard scenarios using invocation of the ECSW's function by the actor, data sending and receiving between ECSW functions, and message sending and receiving between classes. The messages sent and received between the lifelines are relations that show CAs in the use-case diagram, relations that show data sending and receiving in the use-case diagram, and methods of the ECSW's classes. The direction of the messages corresponds to the direction of inductivity in the class diagrams. The inputs for this task are the hazard scenarios and UML system specifications, and the output is the sequence diagrams that describe the details of the hazard scenario.

3.2.5 Clarification of HW and SW components and conditions related to the occurrence of hazards

The task "clarification of HW and SW components and conditions related to the occurrence of hazards" describes the components related to the occurrence of hazards by analyzing the sequence diagrams and clarifying the HOC. The input for this task is the sequence diagrams, and the output is the list of components and HOCs related to the hazard. The conditions for executing each method in the sequence diagrams are clarified. These consist of conditions for invoking a method and conditions representing inner statuses and data. These conditions are HOCs. Fig.5 shows an example of clarification of the components related to a hazard and HOCs.

3.2.6 Planning HPC for each component

The task "planning hazard prevention countermeasures for each component" analyzes the HOC and plans the HPCs. The inputs are components and HOC related to the hazard, and the output is the HPC. Since hazards occur when executing the components, HOCs are "particular components executed under particular conditions." Therefore, HPC is "not to occur the particular HOC" or "not to invoke the components when occurring the particular HOC." Table 1 shows the list of standard HPCs that are developed by analyzing and summarizing the existing HPCs of the ECSW. HPCs for ECSW are decided by applying the countermeasure policies in Table 1 to the HOC of each component.

Table1 Standard hazard prevention countermeasures for HOPs

HOC		Countermeasure policy	Standard hazard prevention countermeasures
Prevention of occurence for specific condition (HOC)	Execution of method	The execution conditions for method are reconsidered.	Review the execution conditions
			Conduct multiple checks when execute
			Conduct the execution check
		The not execution condition of method are reconsidered.	Review the not execution conditions for functions
			Conduct multiple checks when not execute
			Conduct not execution check
	Input/Output	Instructions on Standard Operation Procedure (SOP) misread	Conduct multiple checks on SOP
			Improve the visibility of SOP indications
		Indications on Human Machine Interface (HMI) misread	Conduct multiple checks on HMI
			Improve the visibility of HMI
			Check the content of HMI
		Input data error	Multiple checks on input data
Unexpected events	CPU Load	Unexpected data update occurs	Realize faster processing
			Develop faster devices
		The upper limit of calculation precision is confirmed	Increase significant digits
		The lower limit of calculation precision is confirmed	Increase significant digits
		Divided by zero	Give a warning of division by zero
		Unexpected amount of data is accepted	Refuse data
			Do not input data
		Unexpected interruption tasks occur	Restrict interruption tasks
			Prohibit interruption tasks
		Unexpected CPU load occurs	Unexpected execution requests are not sent
			Refuse unexpected execution requests

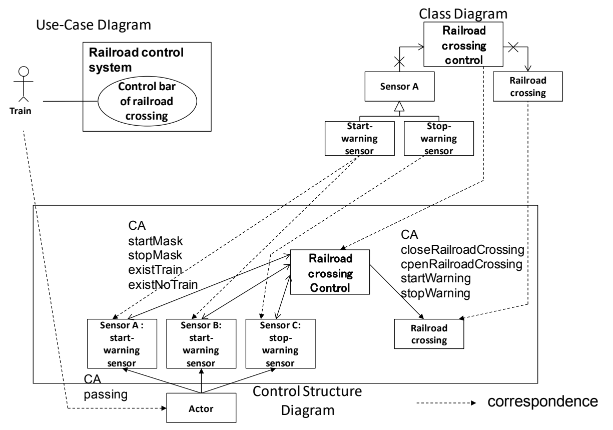

Fig.4 Correspondence between the UML System Specifications and CSD

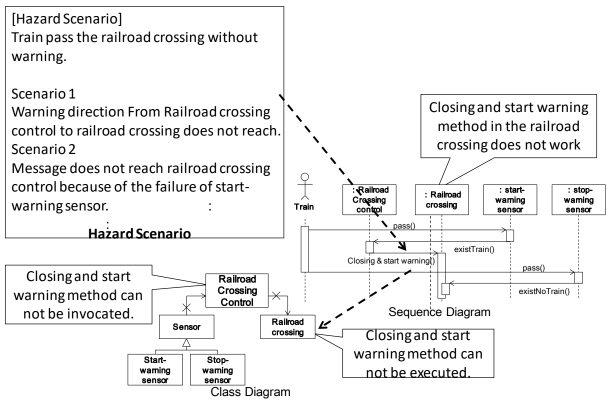
[Hazard Scenario]
Train pass the railroad crossing without warning.
Scenario 1
Warning direction From Railroad crossing control to railroad crossing does not reach.
Scenario 2
Message does not reach railroad crossing control because of the failure of start-warning sensor.
Hazard Scenario
Closing and 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
Class Diagram
Closing and start warning method can not be executed.

Fig.5 An example of clarification of components and HOC related to a hazard

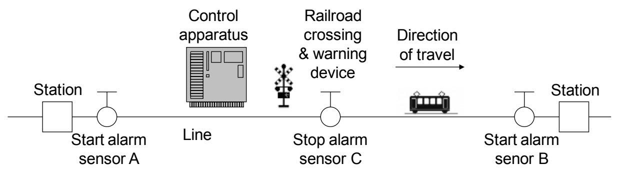

Fig.6 Outline of the railroad crossing control system

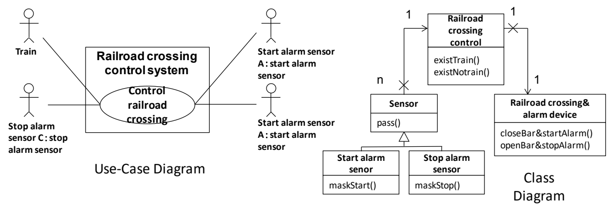
Train
Railroad crossing control system
Control railroad crossing
Stop alarm sensor C : stop alarm sensor
Start alarm sensor A : start alarm sensor
Start alarm sensor A : start alarm sensor
Use-Case Diagram
1
1
Railroad crossing control
existTrain()
existNotrain()
n
1
Sensor
pass()
Railroad crossing&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

Fig.7 Configuration of the railroad crossing control system

4. APPLICATION AND EVALUATION OF THE PROPOSED METHOD

Safety analysis for the railroad crossing control system (RCCS) is conducted to evaluate the method.

First, an outline of the RCCS is the same as the system written in [8]. Fig.6 shows the outline of the RCCS. RCCS consists of the control equipment, the railroad crossing and warning device, and sensors (two warning start sensors, A and B, and one warning stop

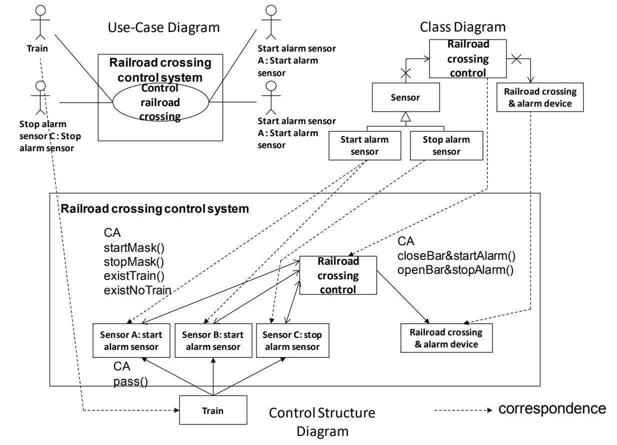
Train
Stop alarm sensor C : Stop alarm sensor
Use-Case Diagram
Railroad crossing control system
Control railroad crossing
Start alarm sensor A : Start alarm sensor
Start alarm sensor A : Start alarm sensor
Class Diagram
Railroad crossing control
Sensor
Railroad crossing & alarm device
Start alarm sensor
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

Fig.8 CSD of the railroad crossing control system

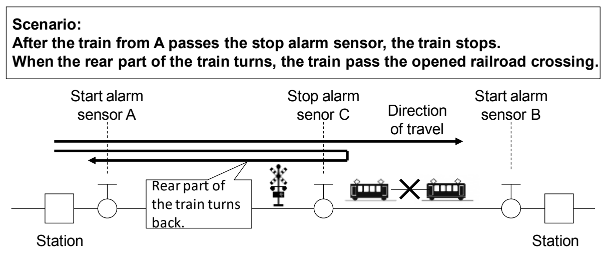
Scenario:
After the train from A passes the stop alarm sensor, the train stops.
When the rear part of the train turns, the train pass the opened railroad crossing.

Fig.9 Hazard Scenario for the target system

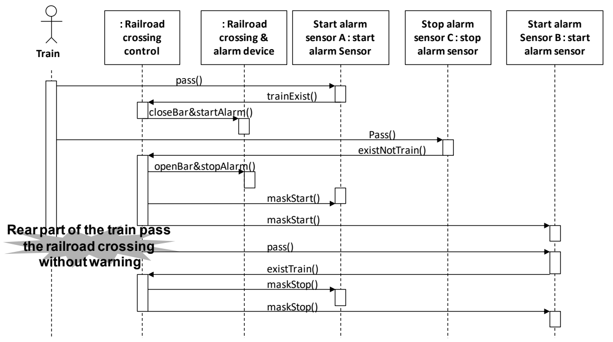
Train
: Railroad crossing control
: Railroad crossing & alarm device
Start alarm sensor A : start alarm Sensor
Stop alarm sensor C : stop alarm sensor
Start alarm Sensor B : start alarm sensor
pass()
trainExist()
closeBar&startAlarm()
Pass()
existNotTrain()
openBar&stopAlarm()
maskStart()
maskStart()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

Fig.10 Sequence diagram corresponding to a hazard scenario

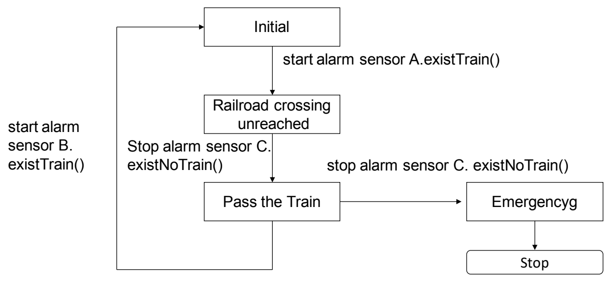
Initial
start alarm sensor A.existTrain()
Railroad crossing unreached
start alarm sensor B. existTrain()
Stop alarm sensor C. existNoTrain()
stop alarm sensor C. existNoTrain()
Pass the Train
Emergencyg
Stop

Fig.11 State machine diagram for railroad crossing control class

sensor, C. These sensors can't detect the direction of the train.) The following are the requirements for RCCS:

Table 2 List of extracted UCAs

	Control Action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order 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.	(UCA2)The train arrives the railroad crossing before rumbling the warning. (closing bar is too late.) (SC1 violation)
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 warnin stops before the train pssses the railroad crossing. (it is too early to open the bar of the railroad crossing after closing the bar.) (SC2 violation)
3	startMask method (Mask enable)	When the train that passes A and c arrives B, the warning rubles again.	(UCA4)When the train does not arrive, the 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)
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 instruction) (SC1 violation)	The warning rumbles again.	Issuing the stopMask instruction start rumbling again, before the train passes B.

Table 3 Hazard scenarios derived from UCAs and guide words

	①Control input or external information wrong or missing	②Inappropriate, ineffective or missing control action	③Delayed operation	④Process input missing or wrong	⑤Unidentified or out-of-range disturbance
(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 of 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 the warning. (closing the bar is too late.)			・Delay of the warning device.		
(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 disturbance before the train arrives the railroad crossing after the 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.			
(UCA5) the warning does not rumble when the train comes because of the delay of issuring the maskStop instruction.			・Delay of issuring the mask stop instruction with the no support for the high spped train.	・Disturbance by the obstacle on the rail.	
(UCA6)The maskStart instruction issues too late.	・Inappropriate external input (disturbance) causes missing of stopMask instruction.	・The delay of issuring the instruction for the control apparatus causes missing the stopMask instruction.	・Inappropriate state control causes the missing the maskStop instruction.	・Inappropriate external input causes the missing of maskStop instruction.	

(1) When the ECSW detects a train using the sensors A or B, it starts warning after a period of time.
(2) When the ECSW detects a train using the sensor C, it stops warning after a period of time.
(3) When the train moves from A to C, sensor B is masked.
(4) When the train moves from B to C, sensor A is masked.

Next, the results of the hazard analysis are explained The task “definition of accidents and hazards” defines hazards and accidents. In this example, an accident is “a train and car collision at the railroad crossing,” and a hazard is “not to be able to close the railroad crossing when the train is in an area.” The task “development of UML system specifications” develops UML system specifications. Fig.7 shows the outline of the RCCS that the authors designed. The use-case diagram shows that the train actor and the sensor actors use the RCSS. The class diagram consists of the RCSS class, sensor class, and railroad crossing and alarm device. The task “analysis of unsafe control actions and development of hazard scenarios using STPA” develops the CSD, clarifies UCAs, and develops hazard scenarios. Fig.8 shows the CSD of the RCCS. UCAs are extracted by applying “the 4 keywords to identify the UCAs that cause hazards” to the CSD. Table 2 shows a list of the UCAs. In addition, the hazard scenarios are derived by applying “the 13 guide words”. Table 3 shows the list of derived hazard scenarios. Subsequently, the case of “inappropriate control of the train when it turns back to the railroad crossing after passing the railroad crossing causes hazards” are analyzed. Fig.9 shows the hazard scenario at this point. The task "development of sequence diagrams corresponding to the hazard scenario" develops sequence diagrams corresponding to the hazard scenario. Fig.10 shows sequence diagrams corresponding to the hazard scenario. The task "clarification of the HW and SW components and conditions related to the occurrence of a hazard" clarifies the components related to the hazard. In Fig.10, after the train passes the sensor C, the RCCS masks the sensors A and B. Afterwards, the rear part of the train turns back and passes the sensor C. At this point, because the status of the crossing is open and the status of the warning device is stopping, the crossing and warning device do not act even if the RCCS issues the open instruction to the crossing and rumbling instruction to the warning device. Therefore, the train enters the railroad crossing when the status of the crossing is open and the status of the warning device is stopping, thereby leading to hazard. As a result, the components related to the hazard scenario are existTrain() and existNoTrain() of the railroad crossing control class. The task "planning hazard prevention countermeasures for the components" plans HPC for each component referring to the standard HPC. The existTrain() method invokes closeBar&start Aram() method of the railroad crossing class. Even if this method is invoked, the bar of the railroad crossing is still closed, and only the warning device is sounded. Therefore, as there is low possibility of this hazard occurring, the countermeasures for this event are not applied. In contrast, the existNoTrain() method invokes openBar&stopAlarm() method in the railroad crossing control class. Generally, the existTrain() method and existNoTrain() method should be carried out in pairs. In addition, the existTrain() method and existNoTrain() method should be invoked alternately. Therefore, "the execution conditions for the method are reconsidered" of HOC in Table 2 is applied. In addition, the "conduct execution check" of the standard HPC is applied. In this case, the state transition diagram shown in Fig.11 is added to the railroad crossing class. In case the existNoTrain() message is received when the state is waiting for the train to pass, countermeasures, such as issuing emergency message to the safety supervisor (the method that issues the warning is added to the railroad crossing class) and sounding the warning are taken. These countermeasures prevent the occurrence of a hazard. As a result of applying the proposed method, hazards associated with the RCCS could be clarified, and appropriate countermeasures to avoid these hazards could be found.

5.FUTURE WORKS

This paper proposed an STPA method applied to hazard analysis for ECSW, developed based on the object-oriented design methodology. It clarifies the causes of hazards that arise from the interactions between a system's components and develops a safe ECSW. However, it was observed that the proposed method requires a long time to analyze hazards and plan countermeasures. In particular, conflicts may arise between countermeasures chosen by the proposed method when analyzing hazards in complex systems because the system includes several components with several hazards and hazard scenarios. In the future, we will investigate a method that describes SCs using logical expressions and analyzes them automatically using logical calculations.

ACKNOWLEDGMENTS

This research was supported by the Scientific Research Grant of the SUZUKI foundation "A Safety Analysis Method Cooperating FMEA, FTA, and HAZOP for Embedded Control Software" and Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science "Integrated Analysis Method for hazard cased by software interaction cooperating with multiple safety analysis methods."

REFERENCES

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

[2] M. Takahashi, R. Nanba, and Y. Fukue, "A Proposal of Operational Risk Management Method Using FMEA for Drug Manufacturing Computerized System", Transaction of the Society of Instrument and Control Engineers, 2012, Vol.48, No.5, pp.285-294.

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

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

[5] N.Leveson, S. Cha, and T. Shineall, "Safety verification of Ada Programs Using Software Fault Trees", IEEE Software, 1991, Vol.8, Issue4, pp.48-59.

[6] M. Takahashi, and R. Nanba, "A Proposal of Fault Tree Analysis for Control Programs", Proc. of SICE Annual Conference 2014, 2014, pp.1719-1724.

[7] G. Pai and J. Dugan, "Automatic Synthesis of Dynamic Fault Tree from UML System Model", Proc. of 13th International Symposium on Software Reliability Engineering, 2002, no page number.

[8] Information-technology Promotion Agency, The first step of STAMP/STPA - A New Safety Analysis Method based on the System Oriented Thinking -, Information-technology Promotion Agency, 2016.