Hazard Analysis for Safety-critical Systems using SOFL

Azma binti Abdullah
Faculty of Computer and Information Sciences
Hosei University, Japan
Faculty of Computer Systems and Software Engineering,
University Malaysia Pahang, Malaysia
azma@ump.edu.my

Shaoying Liu
Faculty of Computer and Information Sciences
Hosei University, Japan
sliu@hosei.ac.jp

***Abstract*— The most important mechanism for improving the safety of a system is to identify the hazard state of the system as it has the potential to cause an unexpected, unplanned or undesired event or a series of events. A hazard that occurs in a system could inevitably lead to an accident (loss event), which could result in an injury or illness or even loss of a human life, and the hazard could also have a negative impact on the environment. An approach in hazard analysis is proposed in this paper in order to avoid hazard from happening in a safety-critical system. The approach consists of three steps: (1) deriving hazards from safety properties, (2) using Fault Tree Analysis (FTA) to analyze the possible causes of each hazard, and (3) converting each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification. A case study based on an Auto-cruise Control (ACC) system for vehicles is used as an example to illustrate the process.**

Keywords— ***hazard; hazard identification; hazard analysis; safety-critical systems***

I. Introduction

Hazard analysis is an examination of a system or subsystems to identify and classify each potential hazard that could occur in the system, and it must be carried out at an early stage of the system development. The aim of the analysis is to deliver a system which does not pose an unacceptable danger to its end-user or to the environment in which the system is installed [1][2]. This analysis can be carried out using a range of techniques, each technique providing a different insight into the characteristics of the system under investigation [3]. If the hazard analysis is not carried out, the hazard events may occur when the system is put into operation. For example, the smart card reader utilised at a train station may reorganize another card into a smart card, or a commuter could travel to his destination even though the credit available on his card is insufficient.

The proposed approach for hazard analysis involves three steps: (1) Deriving hazards from safety properties, (2) Using Fault Tree Analysis (FTA) to analyze the possible causes of each hazard, and (3) Converting each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification. The proposed approach provides guidelines on how to identify hazards, analyze the causes of a hazard and write these causes in formal language for verification. The requirement specification is written in SOFL (Structured Object-Oriented Formal Language) as it provides a formal and comprehensible language for the requirements and design specification as well as a practical method for developing the software systems.

This paper is organized as follows: Section II reviews the background of the research and is followed with Section III which presents the process of hazard analysis. Then Section IV illustrates the proposed process with a case study on an Auto-cruise Control (ACC) system, while Section V describes the related study, and finally, conclusion and recommended future works is described in Section VI.

II. Background

Hazard analysis is a process that is conducted after the safety properties of a system has been captured. A range of techniques can be used in the analysis; each technique has its own style of doing the analysis. In this research, Fault Tree Analysis (FTA) is chosen for the hazard analysis and SOFL language is used to describe the specification module. As such, a brief introduction of SOFL, FTA and some previous studies are hereby presented.

A. SOFL

SOFL (Structured Object-oriented Language), a formal engineering method, is an integration of data flow diagrams, Petri nets and VDM-SL. It provides a formal graphical and textural notation to construct specifications, integrates abstract data types and high-level languages for system implementation, and provides a flexible notation for using structured methods or object-based methods, when appropriate.

A SOFL specification is a hierarchy of condition data flow diagrams (CDFD), where each CDFD in the hierarchy is associated with a specification module. The modules, the most important component of the SOFL specifications, are mainly used to express the architecture of the system. Conceptually a module has the following structure: module name, CDFD and specification of the components, as shows in Fig.1.

Based on Fig.1, the hierarchy of CDFDs and module consists of two CDFDs and the associated modules. Each small rectangle in the CDFDs denotes a process, and each directed line represented a data flow. The CDFD involving processes 1.1, 1.2 and 1.3, corresponding to module SYSTEM, is the top level CDFD. A process performs an action, task, or operation that takes in an input and produces an output; the data flow indicates a data item moving from one process to another; and a data store represents a data repository. A complex process can be decomposed into the subsequent lower level CDFD with an associated module that must be written after the keyword Decom for good traceability in the documentation. As an example based on Fig. 1, process 1.3 is decomposed into the CDFD containing process 1.3.1, 1.3.2 and 1.3.3, and its associated module, named 1.3_Decom [4].

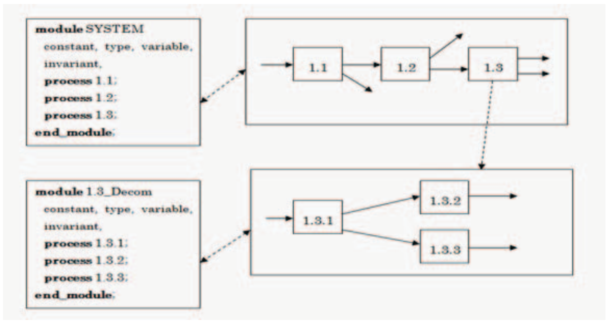


Fig. 1. An Outline of a Specification in SOFL

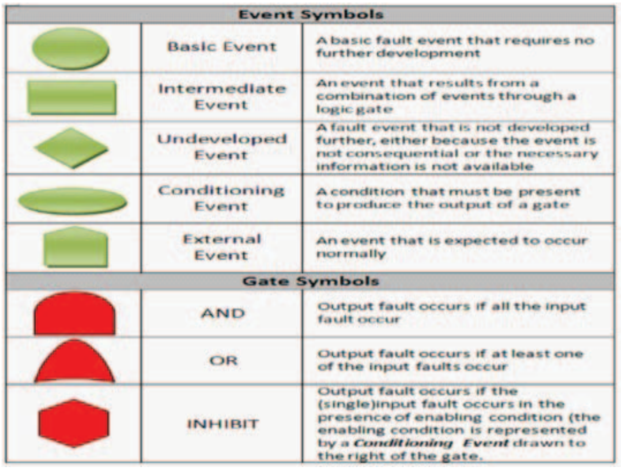
B. Fault Tree Analysis(FTA)

Numerous model and techniques can be used to perform hazard analysis, and this includes FTA[20], checklists[3], event tree analysis[23], failure modes and effect analysis (FMEA)[3][21], and failure modes, effects and criticality analysis (FMECA)[22]. In this research, FTA is chosen for performing the hazard analysis.

FTA was originally developed in 1961 by H.A. Watson of Bell Telephone Laboratories to facilitate the reliability analysis of the Minuteman missile system. FTA is a common and valuable safety analysis technique that is applied during the design phase. It is initially used to analyze the causes of hazards, not to identify the hazards itself.

The fault tree is developed in a top-down manner, and the logic gates indicate the passage of the fault logic up the tree. The event should be traced back until it cannot be developed further due to the lack of knowledge or because no other causes can be identified. The approach is graphical and a fault tree is constructed using the standard symbols, several varieties and extension. In this paper, the fault tree is constructed using only the symbols shown in Fig.2.

When FTA is used to analyse a system, all possible causes that lead to a certain unexpected event must be identified. A fault tree can be constructed in the following way: starting with the top event, identify the first-level contributors that cause the top event and connect the first-level contributors with the top event using a logic gate. Then, identify the second-level contributors that cause the first-level event and connect the second-level contributors with the first-level event using a logic gate. This process is repeated until all the required detail is achieved; the causes on the lowest level are called the basic event. Fig.3 demonstrates the construction of a fault tree.



Event Symbols		
⬭	Basic Event	A basic fault event that requires no further development
▭	Intermediate Event	An event that results from a combination of events through a logic gate
◆	Undeveloped Event	A fault event that is not developed further, either because the event is not consequential or the necessary information is not available
⬬	Conditioning Event	A condition that must be present to produce the output of a gate
⬟	External Event	An event that is expected to occur normally
Gate Symbols		
◓	AND	Output fault occurs if all the input fault occur
⌒	OR	Output fault occurs if at least one of the input faults occur
⬢	INHIBIT	Output fault occurs if the (single)input fault occurs in the presence of enabling condition (the enabling condition is represented by a ***Conditioning Event*** drawn to the right of the gate.

Fig. 2. Fault Tree Symbols

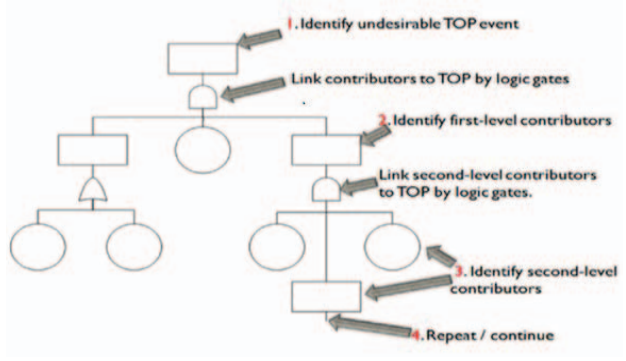

1. Identify undesirable TOP event
Link contributors to TOP by logic gates
2. Identify first-level contributors
Link second-level contributors to TOP by logic gates.
3. Identify second-level contributors
4. Repeat / continue

Fig. 3. Construction of a Fault Tree

The rules for the construction of a fault tree are as follows: (1) all inputs to a logic gate must be described before proceeding to the next gate, (2) each level should be completed before beginning the next level, and (3) start with the direct causes are closest to the top event. The goal of the analysis is to find all minimal cut-sets as it provides the information that helps to identify the weaknesses in the system [6].

C. Previous Work

The study presented in this paper is continuation of the study previously done [7], in which a process to capture the safety properties of a system has been proposed. As shown in Fig.4, the process consists of three steps: (1) Capturing the desired safety-related functions, necessary data resources and constraints, (2) Deriving functional scenario from safety-related function, and (3) Deriving safety properties from functional scenario using the five keys introduces in [7] as guidelines.

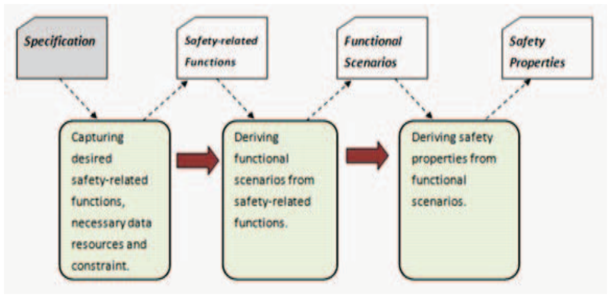

Specification
Safety-related Functions
Functional Scenarios
Safety Properties
Capturing desired safety-related functions, necessary data resources and constraint.
Deriving functional scenarios from safety-related functions.
Deriving safety properties from functional scenarios.

Fig. 4. Capturing Safety property

The five keys for capturing safety properties are functional constraints, domain knowledge, developer experience, real-time constraints for functions and the input/output device. These processes and five keys will assist the developer in finding and identifying the appropriate safety properties required to ensure that the related functions are free from failure. The relationship between capturing safety process and hazard analysis process is shown in Fig.5.

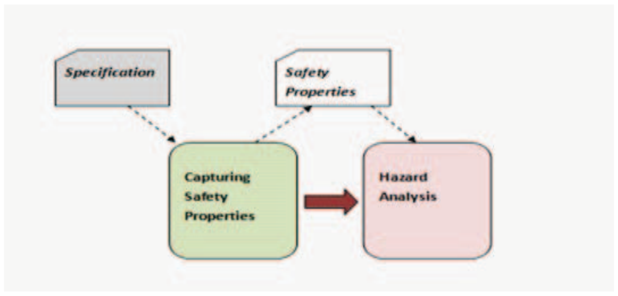


Fig. 5. Relation between Capturing Safety Properties and Hazard Analysis Process

III. THE HAZARD ANALYSIS PROCESS

The process of hazard analysis, as illustrated in Fig.6, consists of three steps: (1) Deriving hazards from safety properties, (2) Using Fault Tree Analysis (FTA) to analyse the possible cause for each hazard, and (3) Converting each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification. Each step that describes an operation is represented by a rounded rectangle box in the figure, and each data item, such as Safety Properties or Hazards, is represented by a card box. An arrow from a card box to an operation means that the data item of the card box is an input to the operation, and an arrow from an operation to a card box means that the data item of card box is an output of the operation. An arrow from one operation to another operation shows a control flow. The detailed explanations of the operations are discussed in the following subsection.

A. Deriving hazards from safety properties

First, hazards for each safety property must be identified. The goal of hazard identification is to recognise the potential failures of system modes that could lead the system to a hazardous state. Gowen et. al [8] described four methods to identify a hazard for safety-critical software systems, which are historical-data analysis, prototype analysis, brainstorming and interviews. Jesty et. al [9] considered the safety policy and safety envelope that needed to be verified to identify a hazard for an automotive system; the safety policy is the company strategy for encouraging and maintaining safety, and the safety envelope is the operational and environmental limits within which the system is expected to behave in a safe manner. John [10] considered data from previous accidents (case study) or operating experience, and scenario development and judgment of knowledge individuals to identify hazard for transportation system. Brady et. al [11] considered ten methods to identify hazards for lunar landing; the methods listed were a combination of human eyes, the human brain, goggles, an external illuminator, a camera and lidar. Numerous methods can be used to identify a hazard; the chosen method depends on the type of critical system to be developed.

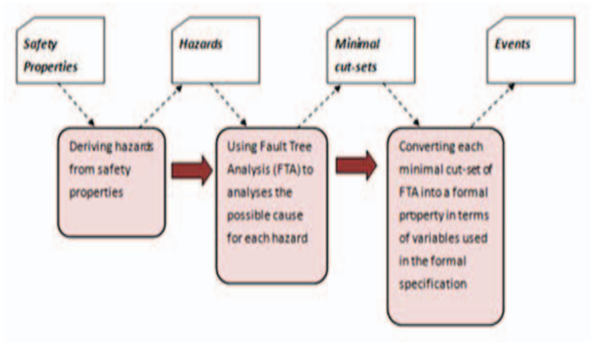

Safety Properties
Hazards
Minimal cut-sets
Events
Deriving hazards from safety properties
Using Fault Tree Analysis (FTA) to analyses the possible cause for each hazard
Converting each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification

Fig. 6. Hazard Analysis Process

B. Using Fault Tree Analysis (FTA) to analyse the possible cause for each hazard

After confirming the hazards for safety properties, these hazards need to be analysed in order to identify its potential causes. In this research, the analysis is conducted using FTA as: (1) it is a useful tool for reliability and safety analysis; (2) it is a top-down approach starting with an undesirable event, called a top event, and all the possible ways that the top event can happen could be determined; and (3) if there is a critical failure mode, all the possible ways that the mode can occur will be discovered [12].

The top event is a potential hazard that has been derived from safety properties. For each hazard, a fault tree need to be developed and an undesired event need to be defined. Then, the event is resolved into its immediate causes; this resolution of events continues until the basic causes are defined. The purpose of contributing the fault tree is to obtain the minimal cut-sets. Each minimal cut-set is the smallest combination of component failures that would cause the top event to occur if the component failures all occur [13].

As an example, in the hazard fault tree shown in Fig.7, the top event is a hazard and would occur when event A, event B and the subsequent events occur. Given a hazard fault tree HT, the top event HAZARD can be determined by {Event A}, {Event C, Event E}, {Event C, Event F} or {Event C, Event G}. By convention, each set of leaf events of these four branches is called a minimal cut-set of the fault tree HT, and the set of all four minimal cut-sets is called the cut-set of HT; namely, the cut-set of HT is {{Event A}, {Event C, Event E}, {Event C, Event F} {Event C, Event G}}.

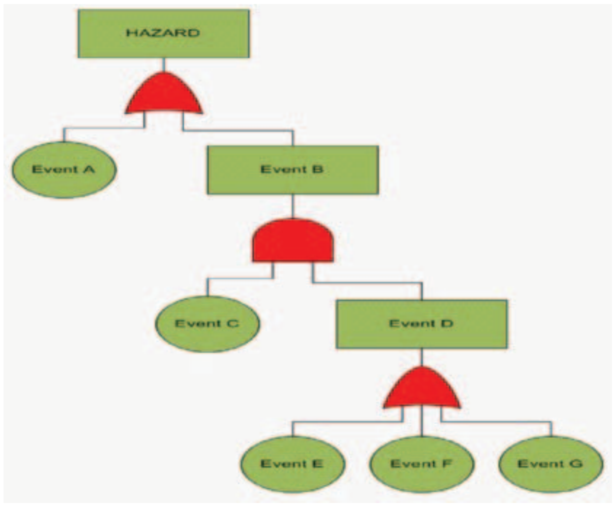


Fig. 7. Fault Tree Analysis for Hazard Analysis

C. *Converting each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification*

The last step is to convert each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification. Fig.8 shows the process that consists of three steps: (1) use the same variable name from the safety-related function; (2) use the same variable name from the module if the variable name from the safety-related function is unsuitable; and (3) create an appropriate variable name if the variable name from the module is still inappropriate. The card box representing the formal specification and minimal cut-set are used for reference. The minimal cut-set is converted to formal property because the result will be used to perform rigorous verification using the testing or inspection techniques. This verification is important to ensure that each property obtained in hazard analysis is not implied by the formal specification.

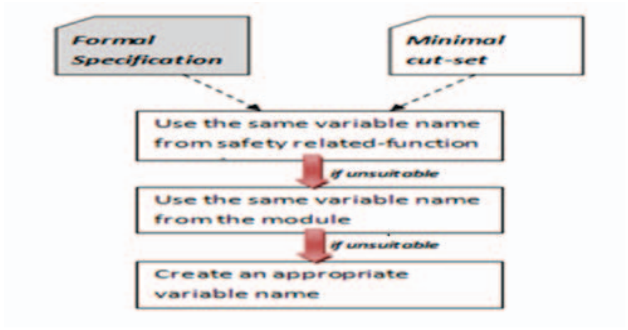
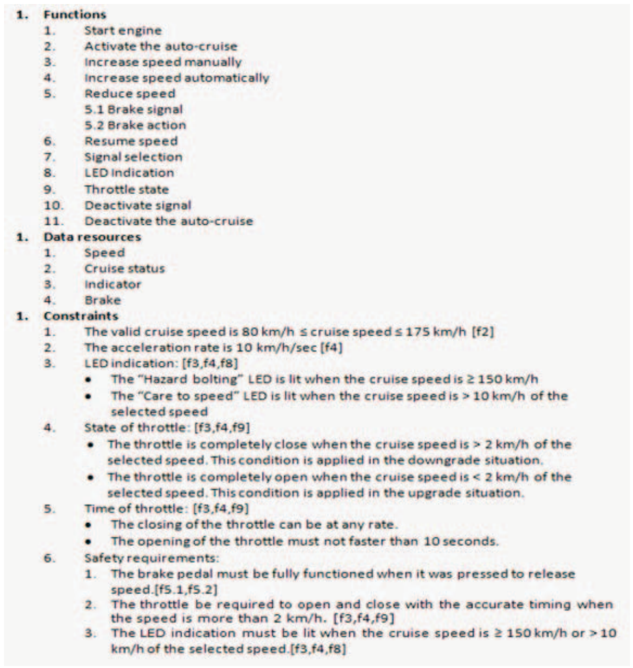

Formal Specification
Minimal cut-set
Use the same variable name from safety related-function
if unsuitable
Use the same variable name from the module
if unsuitable
Create an appropriate variable name

Fig. 8. The Converting Steps

IV. Case Study

In this research, a case study was conducted on Auto-cruise Control (ACC) systems An ACC system is one of the subsystems of on-board automobile (OBA) systems that enhance the safety of the occupants of a vehicle, particularly when travelling over long distances on highways. The ACC system provides the cruise operation with the following four modes: (1) activate or deactivate auto-cruise systems, (2) start or stop acceleration when the vehicle speed is increasing or decreasing, (3) resume control of ACC system, and (4) inform the driver, through LED indication, of certain problems detected while driving the vehicle (for example, via the use of the "Care to speed" LED).



1. **Functions**
   1. Start engine
   2. Activate the auto-cruise
   3. Increase speed manually
   4. Increase speed automatically
   5. Reduce speed
      5.1 Brake signal
      5.2 Brake action
   6. Resume speed
   7. Signal selection
   8. LED Indication
   9. Throttle state
   10. Deactivate signal
   11. Deactivate the auto-cruise
1. **Data resources**
   1. Speed
   2. Cruise status
   3. Indicator
   4. Brake
1. **Constraints**
   1. The valid cruise speed is 80 km/h ≤ cruise speed ≤ 175 km/h [f2]
   2. The acceleration rate is 10 km/h/sec [f4]
   3. LED indication: [f3,f4,f8]
      - The "Hazard bolting" LED is lit when the cruise speed is ≥ 150 km/h
      - The "Care to speed" LED is lit when the cruise speed is > 10 km/h of the selected speed
   4. State of throttle: [f3,f4,f9]
      - The throttle is completely close when the cruise speed is > 2 km/h of the selected speed. This condition is applied in the downgrade situation.
      - The throttle is completely open when the cruise speed is < 2 km/h of the selected speed. This condition is applied in the upgrade situation.
   5. Time of throttle: [f3,f4,f9]
      - The closing of the throttle can be at any rate.
      - The opening of the throttle must not faster than 10 seconds.
   6. Safety requirements:
      1. The brake pedal must be fully functioned when it was pressed to release speed.[f5.1,f5.2]
      2. The throttle be required to open and close with the accurate timing when the speed is more than 2 km/h. [f3,f4,f9]
      3. The LED indication must be lit when the cruise speed is ≥ 150 km/h or > 10 km/h of the selected speed.[f3,f4,f8]

Fig. 9. Informal Specification of ACC Systems

The requirement specification of ACC systems is written using SOFL, and the informal specification for the ACC systems is shown Fig.9. This informal specification consists of three parts: the required functions, the data resources of the ACC systems to be managed, and the constraints reflecting the policy of the auto-cruise. It has eleven functions, and the detail explanation of each function is written in semi-formal specification and formal specification. The semiformal specification is derived from the informal specification, and its goal is to clarify and define all the functions, resources and constraints, and to determine the relations among the three parts contained in the informal specification. The formal specification is made up of the formal implicit specification (abstract design) and the formal explicit specification (detailed design). However, for this research, only the formal implicit specification is used, but the overall picture of the system structure can be seen from the CDFD. The CDFD is a diagram that describes the functional operations, data flows for data communications among processes, and data storage that keeps data for processes to access or update. CDFD for the ACC systems is shown in Fig.10.

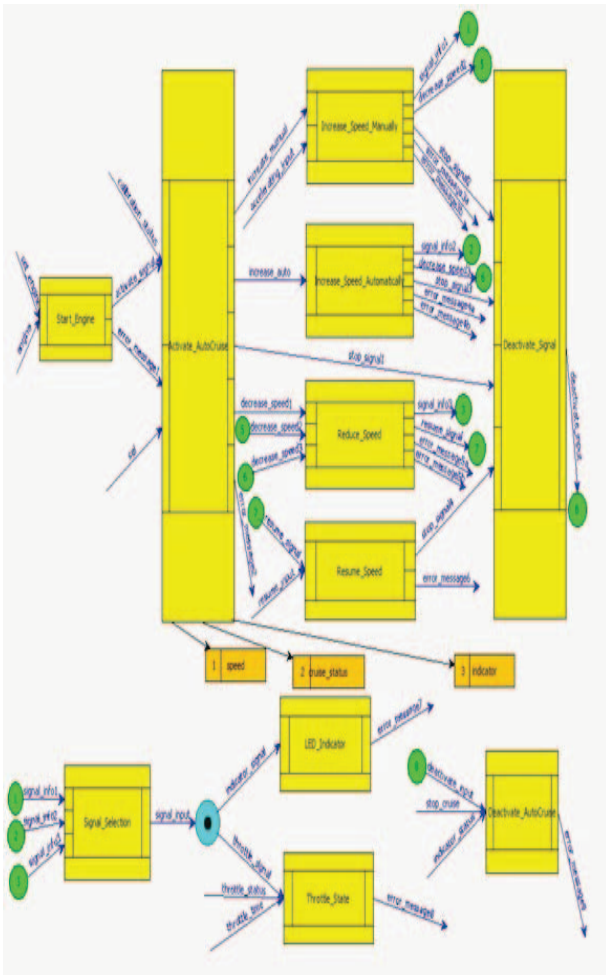


Fig. 10. CDFD of ACC Systems

For this paper, Brake_Action has been chosen as example to illustrate the hazard analysis process. The Brake_Action function is a decomposition from Reduce_Speed function. Fig. 11 shows the CDFD for decomposition of Reduce_Speed function.

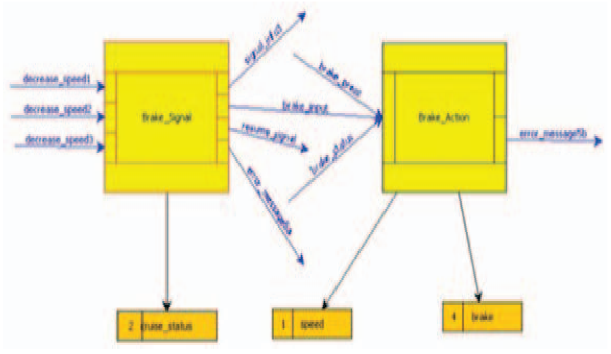
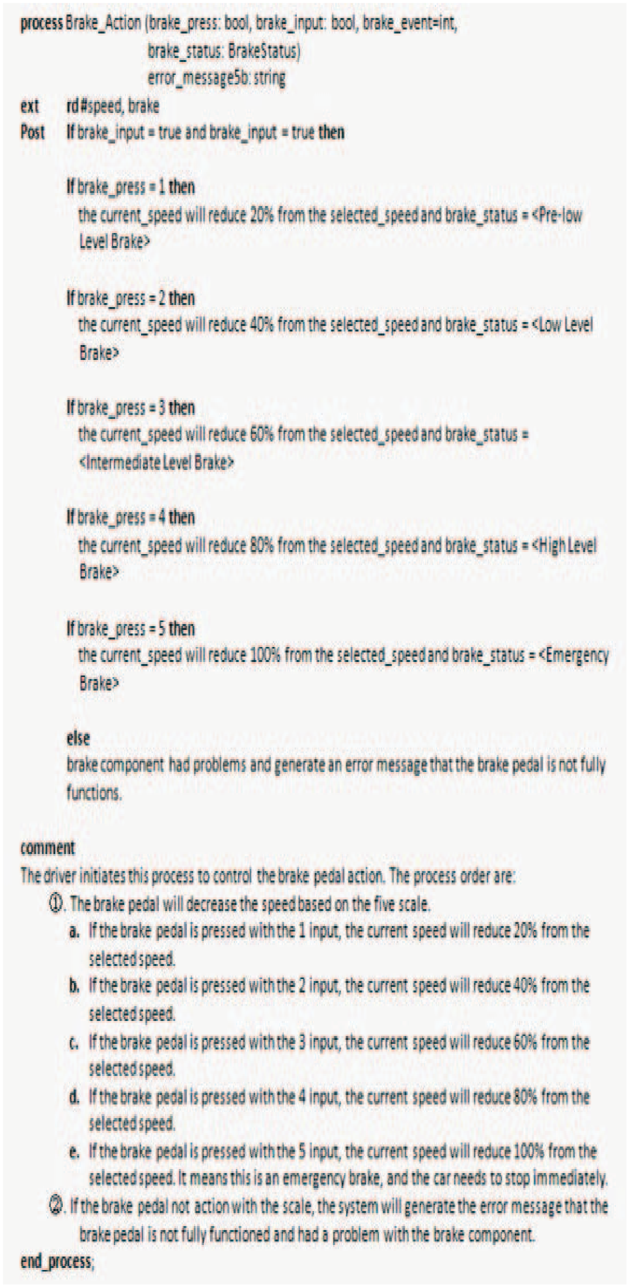


Fig. 11. Semiformal Specification of Brake_Action Function

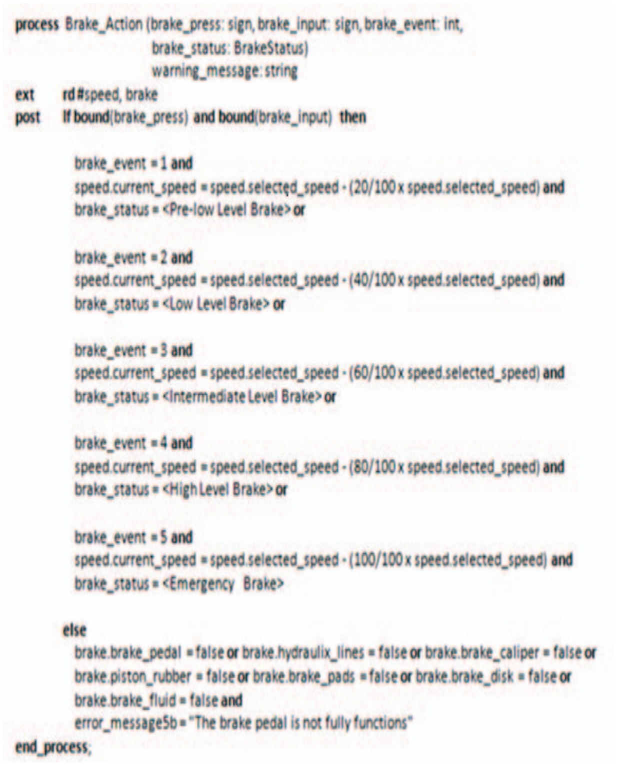
Fig. 12 shows the semiformal specification for Brake_Action. The semiformal specification serves as a channel for communication between the developer and the end user; hence, it should not be fully formal as the user is not expected to understand the specific formalism used in writing the specification.

The formal specification for Brake_Action shows in Fig.13 and the specification is represented in abstract design. The abstract design transforms the semiformal specification into a formal specification that represents the architecture of the entire system and functional definitions of its components.



```
process Brake_Action (brake_press: bool, brake_input: bool, brake_event=int,
                      brake_status: BrakeStatus)
                      error_message5b: string
ext   rd#speed, brake
Post  If brake_input = true and brake_input = true then

      If brake_press = 1 then
        the current_speed will reduce 20% from the selected_speed and brake_status = <Pre-low
        Level Brake>

      If brake_press = 2 then
        the current_speed will reduce 40% from the selected_speed and brake_status = <Low Level
        Brake>

      If brake_press = 3 then
        the current_speed will reduce 60% from the selected_speed and brake_status =
        <Intermediate Level Brake>

      If brake_press = 4 then
        the current_speed will reduce 80% from the selected_speed and brake_status = <High Level
        Brake>

      If brake_press = 5 then
        the current_speed will reduce 100% from the selected_speed and brake_status = <Emergency
        Brake>

      else
      brake component had problems and generate an error message that the brake pedal is not fully
      functions.

comment
The driver initiates this process to control the brake pedal action. The process order are:
   ①. The brake pedal will decrease the speed based on the five scale.
      a. If the brake pedal is pressed with the 1 input, the current speed will reduce 20% from the
         selected speed.
      b. If the brake pedal is pressed with the 2 input, the current speed will reduce 40% from the
         selected speed.
      c. If the brake pedal is pressed with the 3 input, the current speed will reduce 60% from the
         selected speed.
      d. If the brake pedal is pressed with the 4 input, the current speed will reduce 80% from the
         selected speed.
      e. If the brake pedal is pressed with the 5 input, the current speed will reduce 100% from the
         selected speed. It means this is an emergency brake, and the car needs to stop immediately.
   ②. If the brake pedal not action with the scale, the system will generate the error message that the
      brake pedal is not fully functioned and had a problem with the brake component.
end_process;
```

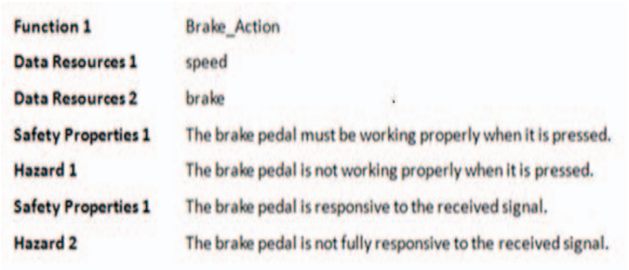
Fig. 12. The Decomposition of Reduce_Speed Function



```
process Brake_Action (brake_press: sign, brake_input: sign, brake_event: int,
                      brake_status: BrakeStatus)
                      warning_message: string
ext     rd #speed, brake
post    If bound(brake_press) and bound(brake_input) then

          brake_event = 1 and
          speed.current_speed = speed.selected_speed - (20/100 x speed.selected_speed) and
          brake_status = <Pre-low Level Brake> or

          brake_event = 2 and
          speed.current_speed = speed.selected_speed - (40/100 x speed.selected_speed) and
          brake_status = <Low Level Brake> or

          brake_event = 3 and
          speed.current_speed = speed.selected_speed - (60/100 x speed.selected_speed) and
          brake_status = <Intermediate Level Brake> or

          brake_event = 4 and
          speed.current_speed = speed.selected_speed - (80/100 x speed.selected_speed) and
          brake_status = <High Level Brake> or

          brake_event = 5 and
          speed.current_speed = speed.selected_speed - (100/100 x speed.selected_speed) and
          brake_status = <Emergency  Brake>

        else
          brake.brake_pedal = false or brake.hydraulix_lines = false or brake.brake_caliper = false or
          brake.piston_rubber = false or brake.brake_pads = false or brake.brake_disk = false or
          brake.brake_fluid = false and
          error_message5b = "The brake pedal is not fully functions"
end_process;
```

Fig. 13. Formal Specification for the Brake_Action Function

A. Deriving hazards from safety properties



Function 1	Brake_Action
Data Resources 1	speed
Data Resources 2	brake
Safety Properties 1	The brake pedal must be working properly when it is pressed.
Hazard 1	The brake pedal is not working properly when it is pressed.
Safety Properties 1	The brake pedal is responsive to the received signal.
Hazard 2	The brake pedal is not fully responsive to the received signal.

Fig. 14. Hazard for the Brake_Action Function

Each safety property may have one or more hazards. For Brake_Action function, the hazards are identified using the historical-data analysis and John.A method. This method identifies hazards by examining the hazards and failures of similar systems, data from previous accidents, and scenario development.

Fig. 14 shows the hazards of the Brake_Action function. The form for hazard has four parts: the function name, data resources, safety properties and hazards of the safety-related function. For this function, there are two safety properties and two hazards.

B. Using Fault Tree Analysis (FTA) to analyse the possible cause for each hazard

After confirming the hazards for each safety properties, each hazards must be analysed, and in this paper, Hazard 1, which is "The brake pedal is not working properly when it is pressed", has been chosen as an example. Fig.15 shows the fault tree of Hazard 1, and there are four minimal cut-sets or events: (1) the brake component did not meet the industrial standard, (2) the hydraulic lines were not properly joined, (3) the piston rubber failed, and (4) the brake fluid contained air and water.

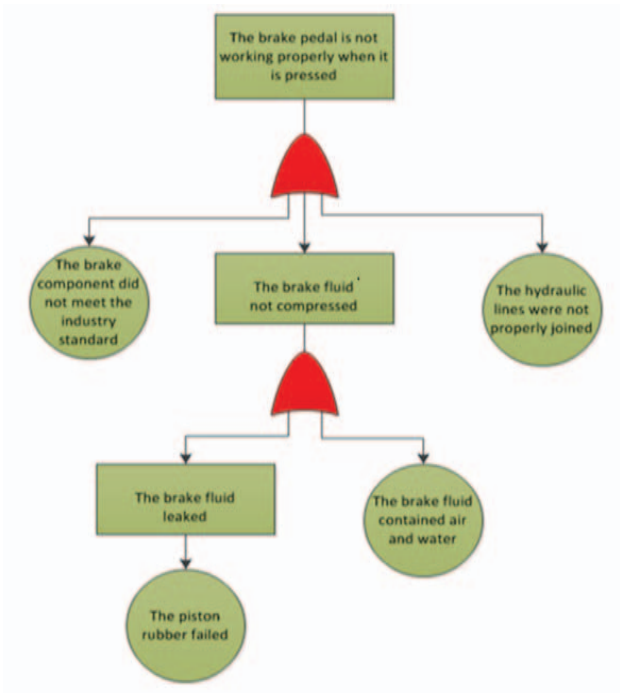

The brake pedal is not working properly when it is pressed
The brake component did not meet the industry standard
The brake fluid not compressed
The hydraulic lines were not properly joined
The brake fluid leaked
The brake fluid contained air and water
The piston rubber failed

Fig. 15. Fault Tree Analysis (FTA) for the Hazard One

C. Converting each minimal cut-set of FTA into a formal property in terms of variables used in the formal specification

The last step is to convert each minimal cut-set to formal language. Fig.16 shows the minimal cut-set of Hazard 1, and the same variables from the formal specification are used to represent this minimal cut-set in formal property. For example, when event number 4 “The brake fluid contained air and water” is converted to formal property, it represents “brake_fluid=false”.

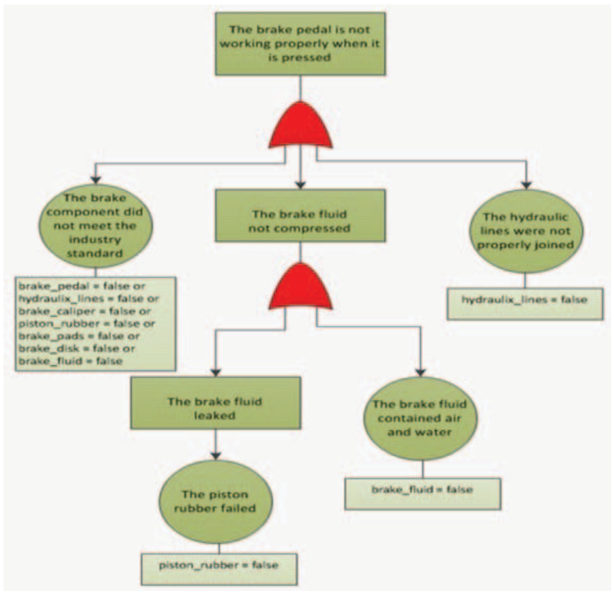

The brake pedal is not working properly when it is pressed
The brake component did not meet the industry standard
brake_pedal = false or
hydraulix_lines = false or
brake_caliper = false or
piston_rubber = false or
brake_pads = false or
brake_disk = false or
brake_fluid = false
The brake fluid not compressed
The hydraulic lines were not properly joined
hydraulix_lines = false
The brake fluid leaked
The brake fluid contained air and water
brake_fluid = false
The piston rubber failed
piston_rubber = false

Fig. 16. Formal Language for the Minimal cut-set Brake_Action Function

V. RELATED WORK

Ericson [14] defined four types of hazard analysis: preliminary hazard analysis (PHA), subsystem hazard analysis (SSHA), system/integrated hazard analysis (SHA/IHA), and operating and support hazard analysis (O&SHA). Neil [15] stated that PHA is used as an early means of hazard identification during the design and development processes. PHA consists of five tasks: (1) determine what hazards might exist during the operation of the system and their relative magnitude; (2) develop guidelines, specifications and criteria to be followed in system design; (3) initiate actions to control particular hazards; (4) identify management and technical responsibilities for action and risk acceptance and assure that effective control is exercised over the hazards; and (5) determine the magnitude and complexity of the safety problems in the program (i.e., how much management and engineering attention is required to minimize and control the hazards).

Several research projects have been conducted to improve PHA techniques. Gowen et.al [8] improved the PHA process by providing a framework as a guideline to identify, eliminate, document and review hazards. The framework has seven steps: defining terminology, obtaining system level hazards, identifying software-related hazards, assessing hazard consequences, categorizing hazards, documenting hazards and reviewing hazards. Though the result from this framework is in natural language, it is difficult to verify. Zhao et.al [16] modified the traditional PHA into the reliability- centred preliminary hazard analysis (RCPHA) as the effect from traditional PHA is limited; the method used by Zhao is more concerned with risk assessment activities. Klim [17] used fuzzy logic to add more powerful features to a classic hazard assessment methodology. The risk-ranking matrix was adopted to create the fuzzy set rules, and he claimed that the hazard analysis performed with the fuzzy risk ranking better reflect some real scenarios of the system failures. Each type of hazard analysis is supported by a number of tools and techniques; in addition, the range of techniques provides a different insight into the characteristics of the system under investigation.

One major problem in performing hazard analysis is in selecting an appropriate model and techniques that match the projects goals. In this research, fault tree analysis (FTA) is chosen as a technique to analyse hazards. Several research works combined FTA with a formal method or formal specification languages. Pahsa et. al [18] calculated failures and faults to identify the most critical functions. They calculated the probability of occurrence of the fire hazard from FTA and Petri Net modelling, and the results show that Petri Net modelling is more accurate and more quantitative for the required specifications of the system safety analysis. Xiang et. al [15] transforms FTA to formal FTA by specifying the safety properties found in FTA with CafeOBJ and verify them via a proof structure. CafeOBJ [19] is the language used for writing and verifying formal specifications of models for a wide variety of software and systems. In this research, combination of FTA and SOFL are used to conduct the analysis.

VI. FUTURE WORK AND CONCLUSION

This paper illustrates an approach to deriving and analyzing hazards from safety properties by presenting a case study of an ACC system as an example. The case study has demonstrated that hazards can be derived and analysed in three steps. This paper focuses only on how to derive and analyse hazards from safety properties without considering any critical or complicated issues, such as concurrent hazards. A better verification technique still needs to be explored for future research.

REFERENCES

[1] Gerald Kotonya and Ian Sommerville, “Integrating Safety Analysis and Requirements Engineering,” International Computer Science Conference, 1997.

[2] Clifton A. Ericson, Hazard Analysis Techniques for System Safety,John Wiley & Sons, 2005.

[3] Nancy G. Leveson, Safeware System Safety and Computers, Addison-Wesley, 2001.

[4] Shaoying Liu, Formal Engineering for Industrial Software Development, Springer, 2004.

[5] John Gould, Michael Glossop and Agamemnon Loannides, Review of Hazard identification Techniques, Report on Health and Safety Laboratory, HSL/2005/58,2000.

[6] Jianwem Xiang, Kokichi Futatsugi and Yanxiang He. “Fault Tree and Formal Method in System Analysis”, Fouth International Conference on Computer and Information Technology, 2004.

[7] Azma Abdullah and Shaoying Liu, “Capturing Safety Properties for Safety-critical Systems using SOFL”, International Conference on Computers Networks Systems and Industrial Engineering, 2011.

[8] Lon D. Gowen, James S. Collofello and Frank W. Callis, “Preliminary Hazard Analysis for Safety-critical Software Systems”, Eleventh Annual International Phoenix Conference, 1992.

[9] P. H. Jesty D.D. Ward and Ward and R.S.Rivett, “Hazard Analysis for Programmable Automotive System”, International Conference on Institution of Engineering and Technology System Safety, 2007.

[10] John N. Balog, Annabelle Boyd and James E. Caton. The Public Transportation System Security and Emergency Preparedness Planning Guide, Report on U.S. Department of Transportation, 2003.

[11] Tye Brady, Edward Robertson, Chirold Epp, stephen Paschall and Doug Zimpfer, "Hazard Detection Methods for Lunar landing", IEEE Aerospace Conference, 2009.

[12] Karen Allenby and Tim Kelly, "Deriving Safety Requirements using Scenarios", Fifth IEEE International Symposium on Requirements Engineering, 2001.

[13] Jianwen Xiang, Kokichi Futatsugi and Yanxiang He, "Fault Tree and Formal Method in System Safety Analysis", Fourth International Conference on Computer and Information Technology, 2004.

[14] Clifton A.Ericson. Hazard Analysis Techniques for System Safety, John Wiley & Sons, 2005.

[15] Neil Storey, Safety-critical Computer Systems, Addison-Wesley, 1996.

[16] Nuo Zhao, Tingdi Zhao and Jin Tian, "Realibility Centered Preliminary Hazard Analysis", Realibility and Maintainability Symposium, 2009.

[17] Z.H.Klim,"Preliminary Hazard Analysis for Design Alternatives based on Fuzzy Methodology", IEEE Fuzzy Information Conference, 2004.

[18] Alper Pahsa, T. Arinc Bayazit, Gokcen Alat, and Buyurman Baykal, "Fault Tree Analysis of Fire Hazard of a Power Distribution Cabinet with Petri Nets", IEEE International Conference on Systems Man and Cybernetis, 2010.

[19] Kokichi Futatsugi, "Verifying Specification with Proof Scores in CafeOBJ", IEEE International Conference on Automated Software Engineering, 2006.

[20] Clifton A. Ericson, "Fault Tree Analysis-A History", 17th International Systems Safety Conference, 1999.

[21] Peter L. Goddard, Raytheon and Troy, "Software FMEA Techniques", Annual Reliability and Maintainability Sysmposium, 2000.

[22] James H. Graham, "FMECA Control for Software Development", Computer Software and Applications Conference, 2005.

[23] Alireza Ahmadi and Peter Soderholm, "Assessment of Operational Consequences of Air-craft Failures: Using Event Tree Analysis", IEEE Transactions on Reliability, Volume 49, No.2, June 2000.