An Analysis Method with Failure Scenario Matrix for Specifying Unexpected Obstacles in Embedded Systems

Toshiro Mise
Matsushita Electoric Works System Solutions Co., Ltd.
3-1-24 Yakuin, Chuo-ku, Fukuoka, 810-0022, Japan
mise@qrl.mew.co.jp

Yasufumi Shinyashiki
Matsushita Electoric Works, Ltd.
1048 Kadoma, Osaka, 571-8686, Japan
yasufumi@icrl.mew.co.jp

Masaaki Hashimoto
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, 820-8502, Japan
hasimoto@ci.kyutech.ac.jp

Naoyasu Ubayashi
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, 820-8502, Japan
ubayashi@ai.kyutech.ac.jp

Keiichi Katamine
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, 820-8502, Japan
katamine@ci.kyutech.ac.jp

Takako Nakatani
S-Lagoon Co., Ltd.
4-12-14-102 Ainokawa Ichikawa, 272-0143, Japan
tina@s-lagoon.co.jp

Abstract

This paper describes an analysis method with failure scenario matrix for specifying unexpected obstacles in order to improve the quality of embedded systems. Although embedded software has become increasingly large in scale and complexity, companies are requiring the software to be developed within shorter periods of time. Therefore, the quality of the software is bound to suffer. This problem is one of the most serious concerns in a coming age of ubiquitous embedded systems. In order to improve the quality, it is very important to specify the forbidden behavior of embedded systems. The forbidden behavior of unexpected obstacles is analyzed by using a matrix and scenarios. This paper provides a detailed description of the analysis method used, in particular the cause, phenomenon, and goal in the scenario, relating them to each other by using a matrix.

1. Introduction

In the domain of developing embedded systems, systems software, which is called "embedded software", has become increasingly large in scale and complexity, since the software has to provide sophisticated functions. However, companies are requiring the software to be developed within shorter periods of time. Therefore, software quality problems, especially system safety problems, have been described in journals and newspapers [2, 12]. The problem is one of the most serious concerns in a coming age of ubiquitous embedded systems.

Products such as electrical appliances within which computer systems are embedded, are used by various kinds of users, including children who know nothing about computer systems, in their individual environments. Moreover, the products are required to be used safely for a long time. To satisfy these quality requirements, embedded software has to provide many functions that can handle exceptions[6, 23]. These functions are implemented by 70 % of the size of the embedded software.

In the early stages of the requirement analysis process, the behavior of the system to be developed must be defined. However, specifying forbidden behavior of the system in order to operate the system safely in various environments in the real world, is very important when developing embedded systems, and the forbidden behavior should therefore be previously defined in the software design.

In this paper, software specifications are divided into two parts, "expected specifications" and "unexpected obstacle specifications". The former specifies the behavior usually described in the software operation manual, and can be explicitly defined when the architecture design process is begun. The latter concerns any deviation from the behavior determined by the former. The deviations that can occur are failure, abnormal behavior, and fading of the system hardware, overload, mis-operation and wrong operation in the operation environment, high temperature, radio noise and rain in the natural environment. Of course, the unexpected obstacle specifications must be explicitly defined by the system specifications. However, we call them "unexpected obstacle specifications" throughout the whole process of system development, in order to distinguish them from the expected specifications, because they are sometimes left undefined.

We have been studying a method for analyzing unexpected obstacle specifications by investigating the work of embedded software experts. This method analyzes failure scenarios by using matrixes to which the advantage of the state transition table, such as the constructibility and comprehensibility of the state and event combination resulting in the unexpected obstacles, is applied. We call the matrixes "Failure Scenario Matrixes". Our previous paper related the matrixes to failure scenarios, and indicated that experts find failure scenarios by using typical phenomena of the scenarios[14, 15].

This paper proposes an analysis method which is detailed as follows: the failure scenario goals which are assumed against satisfying the system quality characteristics, are very useful for finding the scenarios. In each failure scenario, the causes, phenomena, and goal are connected by using the failure scenario matrixes. Section 2 makes clear the research challenges. Section 3 describes the analysis method. Section 4 studies an example. Section 5 discusses the research. Finally, section 6 concludes.

2. Research Challenges

As quality requirements for embedded systems, the following non-functional requirements are as important as those for real time processing efficiency: safety against fire, injury, and electric shock accidents, robustness for maintaining the main function at a maximum despite partial failure, reliability without function errors, and failure prevention against the ripple effect of failure. To improve the quality of embedded systems, the usage and operational environment of the systems as well as the systems themselves should be analyzed throughout their whole life cycle. However, the specification analysis methods for embedded systems have been designed mainly for expected specifications. Therefore, the quality of the specification analysis for unexpected obstacles depends entirely on the skill of the engineers doing the analysis.

Although the information about system problems that have ever arisen has been described in the documents by many companies, the documents have not been fully used. The reason why the documents have not been fully used is as follows: if the information about the problems is described in detail, but fragmentarily, non-experts will be unable to assume problems other than those mentioned by the information. Thus, the range of applicability of the detailed information for non-experts to suppose other problems is too narrow. In contrast, only experts of embedded software will be able to understand information about the problems if the information is described abstractly. However, the experts do not need such abstract information because they already know it.

In order to improve the quality of embedded software, possible failures have to be identified in the specification and design phases of the software. However, there can be unexpected scenarios leading to failures other than the expected scenarios. Actually, the unexpected scenarios sometimes lead to failures because the scenarios are not analyzed in the specification and design phases. As illustrated in Fig. 1, if an event arises, the state transfers to another state. Then, we can see a new phenomenon. Thus, the state transition continues. The continuous transition is understood as a scenario. The scenario can result in an undesirable phenomenon as failure. Failures happen when the behavior of the system deviates from the constraint conditions imposed on the system. However, the constraint conditions are hidden behind the complicated combination of behaviors and unexpected obstacles. It would be difficult to find the constraint conditions without clarifying the failure scenarios.

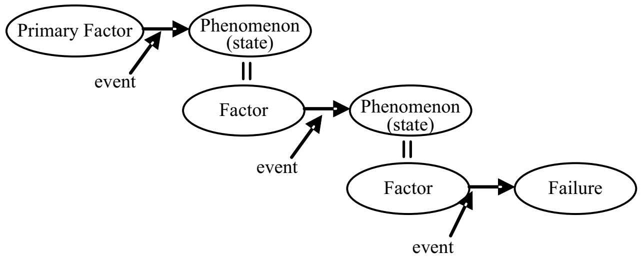


Figure 1. Failure Scenario.

The following two methods were used for analyzing possible failures: FTA (Fault Tree Analysis) and FMEA (Failure Modes and Effect Analysis) [7, 12]. The FTA method analyzes the causes of a failure specified at the root of tree from the root toward the leaves in a stepwise manner. Therefore, the method cannot analyze the causes of failures other than those failures which have been already thought of. The method also does not analyze scenarios and combinations of phenomena. FMEA method analyzes the failures that occur because of problems with system parts [7, 12, 22]. The method does not analyze the scenarios and combinations of phenomena.

We have been being studying two kinds of methods for analyzing embedded software requirements from the viewpoint of unexpected obstacles. One method analyzes them dynamically, although analyzing by this method requires a certain body of expert knowledge [13, 14]. The other method analyzes them statically under some restrictions since analyzing by this method requires a smaller body of expert knowledge [16]. The former method is the one described in this paper. It is a method used for detecting the deficient specifications caused by multiple unexpected obstacles in the phase in which the expected specifications of the system requirements and system architecture such as devices and mechanisms are already obtained by using a tool like UML (Unified Modeling Language) [4, 5, 17, 24, 25, 26, 27].

3. Analysis Method

This section describes the analysis procedure, and then explains the technique used by the embedded software experts.

3.1. Analysis Procedure

The analysis proceeds in the following sequence:

(1) Extracting primary factors of unexpected obstacles

First, the system component objects that have any effect on the system are identified. The effect may consist of giving or modifying the information in the system. The following objects are extracted in order to be analyzed: input circuit, mechanism, target, environment and their construction state as system input, output circuit, mechanism, target, environment and their construction state as system output, function, and memory data as system processing.

The primary unexpected obstacle factors are extracted from the above-mentioned objects. The primary factors specify the states and events which arise directly from the problem's causes such as fading, failure and abnormal behavior of the system parts, contamination, wear down, looseness, breakdown, abnormal positioning, and abnormal behavior of mechanisms.

Next, the unexpected obstacles in the system operation are extracted. The obstacles are unexpected human behaviors such as mis-operation, mis-usage and wrong operation. This analysis should not be done with the aim of developing the system. Rather, it should be done by taking into account what the system's users can do by using the system function and architecture. Moreover, the legal obligations under the Law of Product Liability for foreseeing and avoiding failures, should be met.

The objects listed above and their relationships are described in the system function and architecture diagram. The primary factors are also described in the diagram. By analyzing the diagram, the deficient objects, relationships, and primary factors are checked. In the diagram, a rectangular box specifies an object. Its name is described in the upper part of the box. The names of the individual states that the object can have are written in the lower part of the box. The relationships between the objects are specified by solid lines between the boxes representing the objects. In analyzing the system operation, humans are specified as objects.

(2) Analyzing failure scenario by using failure scenario matrix

In analyzing failure scenarios, the failure scenario matrix is made to include the states and events that are obtained as the above-mentioned primary factors and expected specifications. However, it would be impractical to think of all the states and events that are obtained from every unexpected obstacle, because the number of combinations of states and events is huge. Only the scenarios leading to failures, are required. Therefore, the scope of the states obtained from the unexpected obstacles to be analyzed, is restricted. The states in the scope are analyzed by being combined with the events, whether or not the states result in failure.

Fig.2 illustrates an failure scenario matrix. The states and events obtained from expected specifications and unexpected obstacles are respectively described in the top line and left-most column of the matrix. A rectangular box with three parts is set on each intersection of each line and column. In the top part, a phenomenon that arises when the event specified by the line comes while staying in the state determined by the column is described. The phenomenon specifies a state or event. In the middle part, means to detect the phenomenon is written if the means exist. In the bottom part, means to avoid the phenomenon are described if the means are obtained. If the means cannot avoid the phenomenon perfectly, the phenomenon is added to the top state row as a state or to left-most event column as an event, in order to continue the analysis. The phenomenon can be added to both the row and column as a state and event if it is needed.

3.2. Empirical Techniques of Embedded Software Experts

Sometimes, non-experts of embedded software cannot develop efficiently the failure scenario matrixes from the primary unexpected obstacle factors. On the contrary, experts can analyze a system although they have no experience of it. Therefore, we have analyzed the technique and knowledge of experts for studying a method to help non-experts as follows:

(1) Analyzing failure scenarios

Experts analyze failure scenarios using the following procedure:

a) Quality and possible failures: Embedded systems require different quality characteristics, such as safety, robustness, reliability, and failure controllability, from each

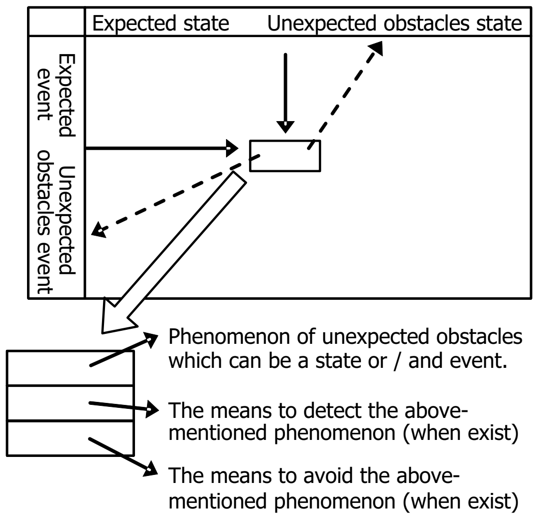
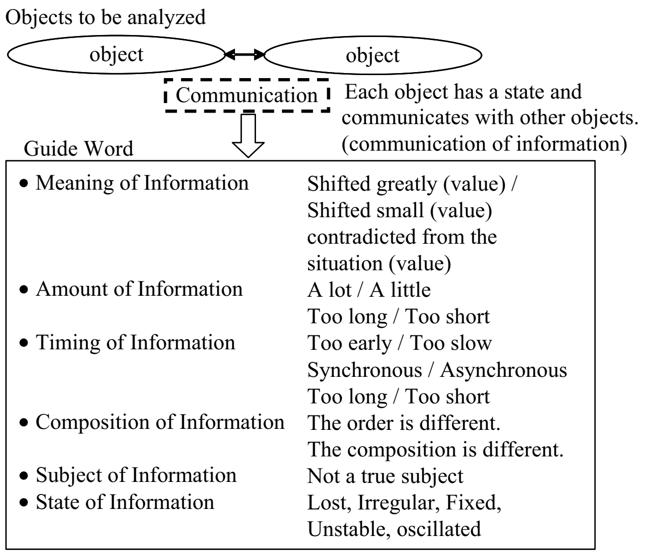

Expected state
Unexpected obstacles state
Expected event
Unexpected obstacles event
Phenomenon of unexpected obstacles which can be a state or / and event.
The means to detect the above-mentioned phenomenon (when exist)
The means to avoid the above-mentioned phenomenon (when exist)

Figure 2. Form of Failure Scenario Matrix.


Objects to be analyzed
object
object
Communication
Each object has a state and communicates with other objects. (communication of information)
Guide Word

• Meaning of Information	Shifted greatly (value) / Shifted small (value) contradicted from the situation (value)
• Amount of Information	A lot / A little Too long / Too short
• Timing of Information	Too early / Too slow Synchronous / Asynchronous Too long / Too short
• Composition of Information	The order is different. The composition is different.
• Subject of Information	Not a true subject
• State of Information	Lost, Irregular, Fixed, Unstable, oscillated

Figure 3. Guide Words for Unexpected Obstacles.

other. Therefore, the quality characteristics are first determined for the system to be developed. To satisfy these quality characteristics, possible failures to be avoided are extracted from the function and structure of the system. For example, fire accidents which are caused by friction heat generated during no-load operation, can be extracted. Of course, the possibility of application specific failures is investigated from points of view such as legal and operational constraints. To analyze the end result failures in detail, the above-mentioned method FTA is applied. Extracting possible failures like this helps to construct failure scenarios.

b) Unexpected obstacle phenomena on failure scenarios: HAZOP (the Hazard and Operability) is a method for analyzing the safety of plants like oil refineries. It assumes deviational phenomena, such as flow stopped and flow increased, by combining the process parameters in the place to be analyzed like volume, pressure, temperature and level of fluid with the guide words such as “increase”, “decrease”, “lack” and “reverse”. Then, it analyzes the impact of the phenomena against the plant, and evaluates the safety [9, 20, 21]. Embedded systems including their operational environment to be analyzed, can be represented as a group of objects and communications between them. The places to be analyzed are arranged on the communications as illustrated in Fig. 3. The guide words concerning the value, timing, form and so on of information are applied to the places in order to suppose the deviational phenomena. They are added to the failure scenario matrix in order to lead the building of failure scenarios as specified in Fig. 4.

(2) Decomposing analysis domain

The larger an embedded system is in scale, the greater its analysis domain is in volume. Therefore, the analysis domain is decomposed in two ways. One is determined by the system architecture which is the hierarchy of hardware, operating systems, middleware and application software. For example, the unexpected obstacles that have arisen as a result of communications noise are analyzed only in the middleware level. The other is decided by a set of service domains. In the architectural level of application software, expected specifications are decomposed into a set of service specifications. For the service specifications, failure scenarios are analyzed. Of course, interactions can happen among the services. Then, the interactions are described as external events of the services and reference to states of other services in the failure scenario matrix.

4. Case Study

SESSAME (the Society of Embedded Software Skill Acquisition for Managers and Engineers) provides requirement specifications for an electric pot as a kind of embedded system [19]. Electric pots are considered a type of consumer goods, and consumer goods should be designed with quality characteristics such as high reliability, durability, safety and fault preventability. Thus, we selected the requirement specifications as a target to reveal the undefined requirements of the pot for satisfying the above-mentioned quality characteristics. This section describes the case study conducted on an electric pot by using the analysis method.

(1) Extracting primary unexpected obstacle factors

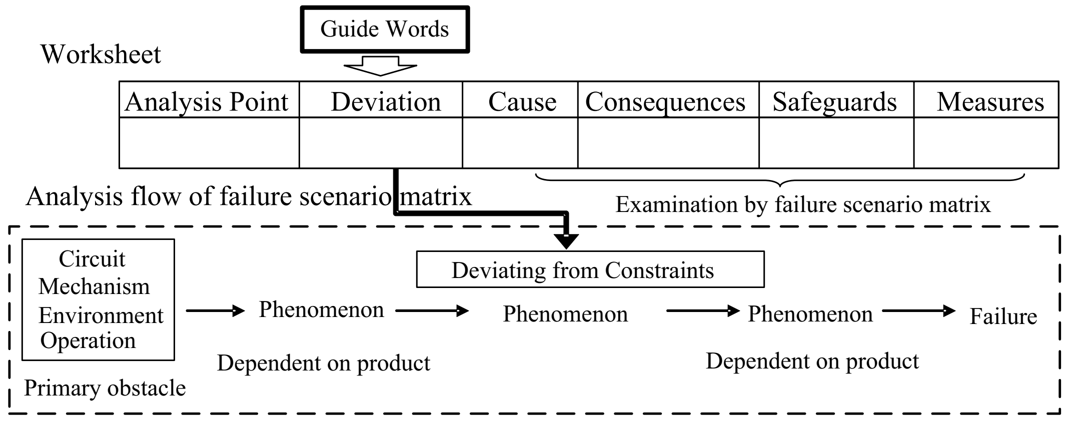


Figure 4. HAZOP Worksheet and Failure Scenario Matrix.

Fig. 5 illustrates partially the function and architecture diagram of the electric pot used in the analysis. First, the water level sensor detects only the volume of water. Then, the functions to observe the increasing rate of water temperature when the heater of the pot is ON and decreasing rate when OFF are extracted to find the cover-open detection sensor failure, pot body tilt and fall down. Moreover, the outside air temperature and pressure are extracted. Concerning the operation, throwing down and shaking the pot, pulling out the electric power cord, pouring alcohol or oil into the pot, and pressing two buttons simultaneously are extracted.

(2) Analyzing the failure scenario by using the failure scenario matrix

The quality characteristics of the pot are analyzed. Then possible failures are extracted as failure scenario goals to be avoided to satisfy the quality characteristics as follows: an accident which would cause a burn can occur against the safety of the pot when using the pot, if boiled water is spilled, or if a fire starts near the heater used to heat the pot. To satisfy the robustness of the pot, an occurrence of a single problem of the cover-open detection sensor or water level detection sensor, should not stop the main function of the pot that heats water. The failure that the water temperature is not adjusted to a specified temperature is extracted against the reliability of the pot.

Next, unexpected obstacle phenomena are extracted. By using the guide words applied to the communications illustrated in Fig. 4, the following deviation phenomena of the water level are extracted: The water level shown by a water level detection sensor suddenly rises. It is caused by falling of the pot with the cover closed, pouring water into the pot, or trouble with the sensor. The water level shown by a water level detection sensor suddenly falls. The cause of the phenomenon is falling of the pot with the cover opened, spilling water out of the pot though the electric power is on, or trouble with the sensor. The water level shown by a water level detection sensor is shaking. Its cause is the pot being shaken or trouble with the sensor.

Then, failure scenarios such as a burn accident by boiling water leaking from an inclining pot and a fire accident by overheating without water are analyzed by using the failure scenario matrix. The examples of the failure scenario matrix and failure scenario of a burn accident are illustrated in Figs 6 and 7 respectively. In the matrix, only the phenomena are specified on the intersections of rows and columns. Their detecting and processing methods are not described in the figure. The analysis procedure is:

a) The end resulting failure "a person is burned by boiling water from the pot" has already been assumed by analyzing the quality characteristic: safety. The failure is written as the end result of the scenario.

b) From the end resulting failure, its cause "the boiled water leaks from the gap between the lid and the body of pot" has already been extracted. The cause is described as the phenomenon preceding the end resulting failure.

c) The exceptional state "the pot inclines" has already been extracted as one of the primary factors. The state is entered in the top state row of the matrix.

d) Then, a normal event "the heater turns on" arises. The event is written in the left-most event column of the matrix.

e) The exceptional phenomenon "the water level detected by the level sensor is not equal to the water level inferred from the rising rate of water temperature." is found from the state and event mentioned in c) and d) respectively. The phenomenon is entered at the intersection of the state column and event row.

f) The phenomenon is written as an event in the left-most event column.

g) The normal state "high level is indicated by the water level detection sensor" is described on the top state row.

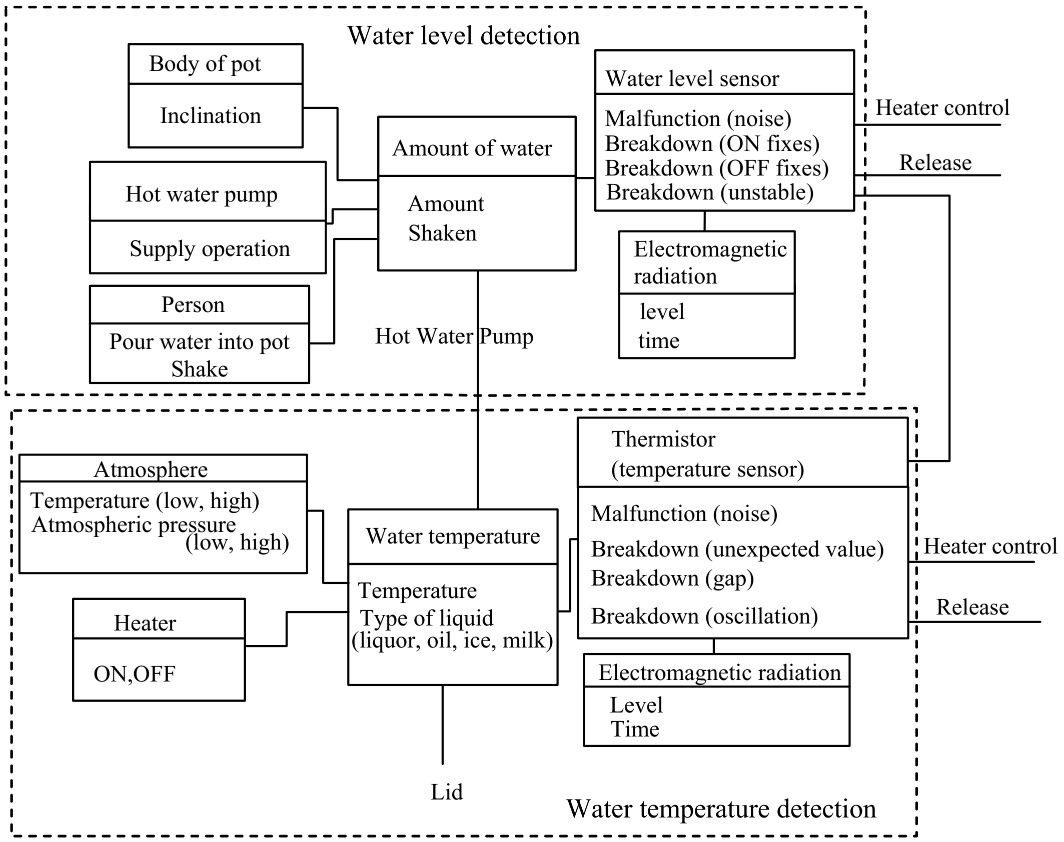


Figure 5. Function and Architecture Diagram.

h) The state and event specified in g) and f) respectively generate the exceptional phenomenon "the water level may get near to the lid although the full water sensor does not detect a full level of water."

i) The phenomenon is written as a state in the top state row.

j) The normal event "the water begins to boil." is specified in the left-most event column.

k) From the state and event described in i) and j) respectively, the exceptional phenomenon of the cause described in b) happens. The phenomenon leads to the end resulting failure described in a).

The failure scenario illustrated in Fig. 7 is built from the above-mentioned analysis. To prevent the failure extracted from the scenario, the heater must not be turned on when the pot reaches the state described in h). Thus, the unexpected obstacle specifications are obtained.

5. Discussion

This section discusses the validity of the method described in this paper.

(1) Comparing with existing methods

From the viewpoint of analyzing unexpected obstacle specifications, FMEA is a method of finding possible troubles and failures of hardware devices. By using the method, the primary factors are obtained. HAZOP is a method for detecting deviations of system parameters. By applying the method, typical phenomena are gained. FTA is a method to find the causes of failures. By using the method, failure scenarios are traced in the reverse direction from resulting failures to the causes. The method described in this paper applies the existing methods as component tools, and combines them by the failure scenario matrixes to find failure scenarios efficiently.

Recently, exception handling is being studied in the domain of requirements engineering. For example, misuse cases as use cases with hostile intent were analyzed by applying the UML method [1, 18]. Obstacle handling was studied in a goal-oriented manner [8]. Abuse frames were used to bound the scope of security problems [3, 10, 11]. These studies adopted a top-down manner in their methods in oder to analyze only serious failures. However, failures of embedded systems are caused by various kinds of factors

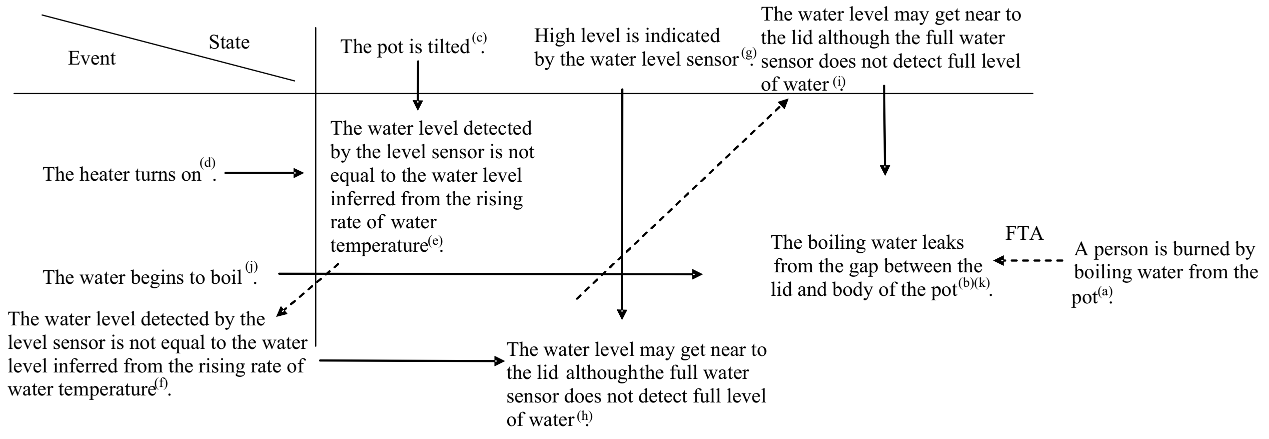

Event
State
The pot is tilted(c).
High level is indicated by the water level sensor(g).
The water level may get near to the lid although the full water sensor does not detect full level of water(i).
The heater turns on(d).
The water level detected by the level sensor is not equal to the water level inferred from the rising rate of water temperature(e).
The water begins to boil(j).
The boiling water leaks from the gap between the lid and body of the pot(b)(k).
FTA
A person is burned by boiling water from the pot(a).
The water level detected by the level sensor is not equal to the water level inferred from the rising rate of water temperature(f).
The water level may get near to the lid although the full water sensor does not detect full level of water(h).

Figure 6. Failure Scenario Matrix for Burn Accident Scenario by Inclining Pot.

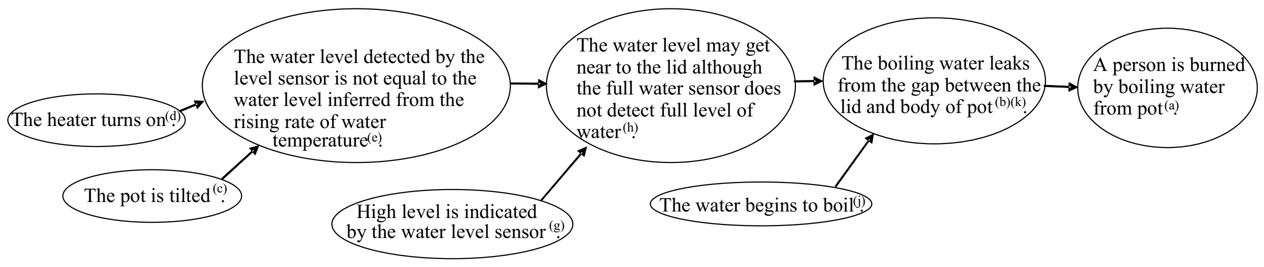

The heater turns on(d).
The pot is tilted(c).
The water level detected by the level sensor is not equal to the water level inferred from the rising rate of water temperature(e).
High level is indicated by the water level sensor(g).
The water level may get near to the lid although the full water sensor does not detect full level of water(h).
The water begins to boil(j).
The boiling water leaks from the gap between the lid and body of pot(b)(k).
A person is burned by boiling water from pot(a).

Figure 7. Burn Accident Scenario by Inclining Pot.

in the hardware, environment and operations. Moreover, slight failures of the systems must be found and analyzed for the simple operations and robustness because even children use the systems. From these points of view, this paper has proposed a failure scenario analysis method which adopted both top-down and bottom-up manners.

(2) Comparing the failure scenario matrix with the state transition table for software design

The state transition table relates exhaustively every event to every state. On contrary, the failure scenario matrix manages only the state and event combination resulting in the unexpected obstacles. Concerning the order of making the state transition table and scenarios, the table is designed on the base of the scenarios. In contrast, the matrix is build prior to the scenarios. The states and events in the table are used for controlling the system. On contrary, the states and events in the matrix restrict the range of thinking for analysts to make the thinking efficient. Thus, the matrix is quite different from the table.

(3) Empirical techniques of embedded software experts

The method described in this paper has been developed by investigating the techniques of experts. Experts have an inexplicit analysis procedure and abstract knowledge. The inexplicit procedure has been made clean in this paper. The abstract knowledge is mainly composed of the viewpoints of classifying failures, phenomena and causes. Since this paper has made a clear distinction between procedure and knowledge, the knowledge could be studied for being formalized.

(4) Future study

Experimental verification of the analysis method described in this paper is required. Prior to the experiment, the above-mentioned knowledge formalization about the classifying viewpoints should be studied. This paper has described the analysis method for unexpected obstacles in the software design process. The method is applicable to the software requirement analysis process. Therefore, we also should study applying the method to requirement analysis process. The above-mentioned knowledge formalization would lead to studying the database of knowledge. Of course, the method described in this paper does not satisfy exhaustive analysis. The limits of the exhaustiveness should be made clear.

6. Conclusion

This paper has described an analysis method for extracting unexpected obstacle specifications by using an failure scenario matrix to which the advantage of the state transition table is applied. We have clarified the following by analyzing the knowledge of embedded software experts: the failure scenario goals which can be determined by referring to the system quality are indispensable for developing the scenarios. The initial causes, resulting phenomena and final goal can be connected using the failure scenario matrix.

The requirement analysis for unexpected obstacle specifications has needed experts in the domain of developing embedded systems. Therefore, the quality and productivity of embedded software have been unstable if the experts have not been assigned to the analysis process. We need to study methods that will decrease the need for experts and make the system quality stable. We also need to study organizational devices for reusing the expert knowledge about unexpected obstacles.

Acknowledgments

The authors would like to thank Masayuki Hirayama and Takeshi Sumi for fruitful discussions.

References

[1] I. Alexander. Misuse cases, use cases with hostile intent. In *IEEE Transactions on Software Engineering*, volume 20(1), pages 22–33, 2003.
[2] T. Aoki. Trend of embedded systems. In *IPSJ SIG Technical Report*, volume 2002-SE-137(9), pages 61–64, 2002.
[3] R. Crook. Security requirements engineering: when anti-requirements hit the fan. In *Proceedings of Requirement Engineering*, pages 22–33, 2002.
[4] B. P. Douglass. *Real Time UML: Advances in the UML for Real–Time Systems*. Addison-Wesley, 2004.
[5] T. Hosokawa, T. Tsurumi, T. Okamoto, and H. Koizumi. A proposal of use–case analysis/design method in embedded software developments. In *IPSJ SIG Technical Report*, volume 2003-SE-140(2), pages 9–14, 2003.
[6] F. Kasati and G. Cugola. *Error Handling in Process Support Systems*, pages 251–270. LNCS 2002, 1998.
[7] K. Kitagawa. *Introduction method of FMEA and FTA*. Research Center, 1984.
[8] A. V. Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. In *IEEE Transactions on Software Engineering, Special Issue on Exception Handling*, volume 26(10), pages 978–1005, 2000.
[9] N. G. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, 1995.
[10] L. Liu, B. Nuseibeh, D. C. Ince, and M. Jackson. Using abuse frames to bound the scope of security problems. In *Proceedings of Requirement Engineering*, pages 354–355, 2004.
[11] L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proceedings of Requirement Engineering*, pages 151–161, 2003.
[12] Ministry of Economy, Trade and Industry, editor. *Report of actual field survey of embedded software*. Ministry of Economy, Trade and Industry, 2004 (In Japanese).
[13] T. Mise, Y. Shinyashiki, M. Hashimoto, N. Ubayashi, K. Katamine, and T. Nakatani. Exception Analysis Matrix for Embedded Systems Software Specification. In *Proceedings of Embedded Software Symposium*, October 2004 (In Japanese).
[14] T. Mise, Y. Shinyashiki, M. Hashimoto, N. Ubayashi, K. Katamine, and T. Nakatani. A Basic Model and Specification Analysis Method for Embedded Software Exceptions. In *IPSJ SIG Technical Report*, volume 147-11, 2005 (In Japanese).
[15] T. Mise, Y. Shinyashiki, M. Hashimoto, N. Ubayashi, K. Katamine, and T. Nakatani. A Specification Analysis Method for Unexpected Obstables in Embedded Software. In *Proceedings of Foundation of Software Engineering 2005*, November 2005 (To be appeared) (In Japanese).
[16] Y. Shinyashiki, T. Mise, Y. Eura, H. Hatanaka, M. Hashimoto, N. Ubayashi, K. Katamine, and T. Nakatani. A Conceptual Model of Exceptions in Embedded Software. In *Proceedings of Embedded Software Symposium*, October 2004.
[17] S. Shlaer and S. J. Mellor. *Object Lifecycles: Modeling the World in States*. Yourdon Press Computing, 1991.
[18] G. Sindre and A. Opdahl. Eliciting security requirements with misuse cases. In *Journal of Requirements Engineering*, volume 10, pages 34–44, 2005.
[19] Society of Embedded Software Skill Acquisition for Managers and Engineers, editor. *Specification of Boiling Jar(GOMA-1015) 3rd edition*. Society of Embedded Software Skill Acquisition for Managers and Engineers, 2003 (In Japanese).
[20] T. Sumi, M. Hirayama, and N. Ubayashi. Analysis of the external environment for embedded systems. In *IPSJ SIG Technical Report*, volume 2004-SE-146(5), pages 33–40, 2003.
[21] T. Sumi, O. Mizuno, T. Kizuno, and M. Hirayama. An effective testing method for hardware related fault in enbedded software. In *IEICE Transactions on Information and Systems*, volume E88-D, pages 1142–1149, June 2005.
[22] J. Suzuki. *Executin method of FMEA and FTA*. JUSE Press, 1998 (In Japanese).
[23] The Institution of Professional Engineers, editor. *Risk Analysis Engineering*. Maruzen, 2004 (In Japanese).
[24] H. Watanabe, M. Watanabe, K. Horimatsu, and K. Tomotake. *Embedded UML*. Shoeisha, 2002.
[25] M. Watanabe. *Enhancing Hierarchy State Transition Table Design Technique Version 2.0*. Cats, 1998 (In Japanese).
[26] M. Watanabe, S. Ishida, K. Asari, S. Iida, and S. Yamamoto. *Embedded System Development by UML Dynamic Model (in Japanese)*. Ohmsha, 2003.
[27] K. E. Wiegers. *Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughtout the Product Development Cycle*. Microsoft Press, 2003.