
28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2024)

Non-Functional Requirements through Exceptions

Shuichiro Yamamoto^{a*}

^a*International Professional University of Technology in Nagoya, 4-27-21 Meieki, Nakamura-ku, Nagoya 450-0002, Japan*

Abstract

In order to prevent information system failures, it is not enough to simply define functional requirements; it is necessary to clearly describe non-functional requirements such as reliability and availability. Diagram Modelling methods have been proposed to integrate non-functional requirements with functional requirements. However, in these methods, the relationship between functional exceptions and non-functional requirements was not clear. In addition, several methods have been proposed to describe non-functional requirements such as performance using functional exceptions. However, the interrelationship between these description methods was not clear.

In this paper, we propose a method for extracting non-functional requirements based on functional exceptions. We also use this method to compare three methods: NFR framework, Control case, and Hatley and Pirbhai Method.

© 2024 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0>)

Peer-review under responsibility of the scientific committee of the 28th International Conference on Knowledge Based and Intelligent information and Engineering Systems

Keywords: Requirements Specification; Non-Functional Requirements; Functional Exceptions; Comparative Study

1. Introduction

As Information systems deployed into society, it is necessary for defining Non-functional requirements to realize sustainable information systems preventing defects that occurs in the system development process.

For this purpose, it is necessary to clarify the method for describing interrelationship between Non-functional requirements and exceptions. So far, many Non-functional requirements method and methods for assuring system safety and security have been proposed. However, it cannot be said that the method of describing relationship between Non-functional requirements and exceptions has sufficiently permeated. If exceptions cannot be expressed in the non-

* Corresponding author. Tel.: +81-52-561-2001; fax.: +81-52-582-0077

E-mail address: yamamoto.shuichiro@gmail.com

functional requirement definition, it is difficult to associate non-functional requirements with functional requirements.

If an exception to a non-functional requirement occurs, the non-functional requirement cannot be satisfied. Conversely, if no exception corresponding to the non-functional requirement occurs, the non-functional requirement is satisfied. Therefore, unless an exception corresponding to the non-functional requirement is made in the functional requirements, the non-functional requirements cannot be realized. In this way, by detecting exceptions to non-functional requirements using functional requests, it is possible to clarify the specific mechanism for associating functional and non-functional requirements.

In this paper, we propose the Non-functional requirements extraction by using exceptions as a model for designing the non-functional requirements analysis process in industry, and clarify that it can be applied to an embedded system of maintainability requirements definition.

Below, Section 2 describes related research. Next, Section 3 proposes Non-functional requirement extraction method by exceptions. Section 4 compares NFR framework, control case, and HPM. Section 5 describes an application example of the proposed method. In Section 6, we discuss our considerations, and in Section 7, we present a summary and future issues.

2. Related work

Related research is explained below.

2.1. Non-Functional Requirements

Robertson and Robertson [1] classify non-functional requirements (NFR) into eight types. That is, appearance requirements, usage requirements, performance requirements, operation requirements, maintenance requirements, security requirements, organizational culture requirements, and legal system requirements.

Appearance requirements describe specifications regarding the appearance of the system, such as its appearance and color usage. Usage requirements describe the level of ease of use of the system from the perspective of the system user.

Performance requirements describe speed, accuracy, capacity, efficiency, and safety as conditions for executing functional requirements. Operational requirements describe the execution conditions that the system's operating environment must satisfy. If the performance of the hardware or OS provided as the execution environment is insufficient, the system will not run properly. Maintenance requests describe the conditions to be met by system changes that occur after operation. For example, there are changes in the operating environment such as the OS, organizational environment, business environment, legal system, etc.

Security requirements describe three aspects of security: confidentiality, integrity, and availability. Organizational culture requirements require naming conventions and culture as an organization's behavioral style to be reflected in business processes. Organizational culture requirements do not have a rational basis, but must be met in order for the system to be used. Legal system requirements must specify the legal system that the system should comply with.

Navarro, Leveson and Lunqvist [2] defined the service quality levels. The service quality levels are management, purpose, design principle, architecture, design specification, physical specification and operation levels. As control models are not explicitly described in the approach, the relationship between functional exceptions and service requirements. Leveson [3] also proposed the completeness criteria of requirements specification that considers unexpected input as state, although the relationship with exceptions hasn't been considered.

2.2. Integration of Non-Functional and Functional Requirements diagrams

Moreira et al. [4] proposed a use case diagram that integrates functional and non-functional requirements using stereotypes that represent non-functional requirements. A use case for a functional request is connected to a use case whose stereotype is a non-functional request through an include relationship.

Losavio [5] extends the use case diagram by using soft goals of NFR framework to describe non-functional requirements and relate them to use cases.

Both approaches use connection lines between nodes of non-functional and functional requirements. It is not clear how non-functional requirements are realized in functional requirements. Moreover, both approaches have not been mentioned exceptions.

2.3. NFR framework

In the NFR framework [6], non-functional requirements are expressed as NFR soft goals. The components of the target architecture are expressed as operational soft goals. Architecture evaluation can be supported by step-by-step elaboration of higher-level NFR soft goals into lower-level NFR soft goals through AND decomposition or OR decomposition, and by associating operational soft goals with the lowest-level NFR soft goals. The SIG (Soft goal Interdependency Graph) is used to represent security and safety goals. The NFR goals are soft goals, operational goals, and claim soft goals. First, constraints of target systems are clarified by NFR soft goals. Soft goals are, then, decomposed into sub goals to develop SIGs. NFR soft goals are allocated to operational soft goals for describing target system functions.

Subramanian and Zalewski [7] proposed a method to evaluate the safety and security of architectures by combining the NFR framework with scenario analysis.

- 1) Decompose safety into NFR soft goals of the NFR framework.
- 2) Decompose security into NFR soft goals in the NFR framework.
- 3) Decompose the architecture into operational soft goals of the NFR framework.
- 4) Define the contribution relationship to the NFR soft goal by the operation soft goal.
- 5) Evaluate safety and security by examining the values of soft goal labels using propagation rules.

Subramanian et al.'s method extracts operational soft goals as functional requirements from non-functional requirements, so the relationship between exceptions to functional requirements and non-functional requirements is not clear.

2.4. Control case

The control case approach proposed by Zou and Pavlovski [8] includes conceptual control cases and detailed control cases. Conceptual control cases describe risks and their effects regarding non-functional requirements such as performance and availability. For example, a detailed control case that describes how an increase in load will delay response time, reduce the number of users, and incur business loss may include system features that eliminate or avoid the risks described in the conceptual control case.

Pavlovski and Zou [9] also applies Control case to business process quality improvement. Since the non-functional characteristics of the business are arguably difficult to capture in business process modeling, they proposed how operating condition and control case may be applied to model the constraints associated with a business process. The operating condition is used to denote a business process constraint and the control case to define controlling criteria to mitigate risk associated with an operational condition. They claimed that the approach contributes to make models more complete representation of the overall business process, because of assisting in mitigating risk and facilitate the early discovery of non-functional requirements during systems development.

2.5. Hatley and Pilbhai Method (HPM)

Hatley and Pirbhai [10] proposed a method to integrate Data flow diagram (DFD) and Control flow diagram (CFD) for describing real time system requirements model. They also proposed AFD (Architecture flow diagram) to design real time system architecture model based on real time requirements models in DFD and CFD.

They clarified the following five points.

- 1) An architecture model can be created in three stages: a technology-independent model that corresponds to the requirements model, a technology-neutral model that takes physical boundaries into account, and a technology-dependent model that corresponds to architecture modules.
- 2) It is necessary to iteratively improve the requirements model and architecture model.
- 3) A real-time system requirement model can be defined using a process model based on data flow diagrams and a

control model based on control flow diagrams.

4) Architecture models can be created by converting the logical boundaries of the requirements model into physical boundaries using architecture templates.

5) An architecture model can be constructed from a diagram consisting of an architecture context diagram, an architecture flow diagram, an architecture interconnection diagram, an architecture module specification, an architecture interconnection specification, and an architecture dictionary.

HPM (Hatley and Pirbhai Method) can be applied to define function exceptions and corresponding functions in the CFD. However, it does not target Non-functional requirements other than performance requirements.

2.6. Business Process Quality

Grigori et.al. [11] proposed a method to improve business process quality through exceptions. They define the high-level user-oriented exception as a deviation from the acceptable process execution that prevents the delivery of services with the desired quality. They tried to achieve business process quality by analyzing, predicting and preventing the occurrence of exceptions based on data warehouse and mining approaches.

In the manufacturing process, there is a misconception that local optimization is necessary, as long as one's own process is fine and that unnecessary problems shall not be introduced to one's own department. If a problem is discovered at the final stage of development, the design cannot be modified or the basic structure of the product cannot be changed. Therefore, comprehensive product design and manufacturing is required throughout the entire production process. JKK (Ji Koutei Kanketsu) [12] is a method that optimizes the entire production process, not just a specific process. The Japanese words Ji, Koutei, and Kanketsu (JKK) correspond to self, process, and completion respectively.

To introduce JKK, it is necessary to define not only business procedures that define the flow of work, but also requirements organization sheets that define business requirements. The requirements organization sheet consists of fields of the necessary items/information, business inputs, and business outputs for each business process. The necessary item and information field clarifies the input, tools, methods, capabilities/authority, and reasons as conditions for the quality of product. The input field describes the receiving criteria, such as when, where, and what. The output field describes where to sink, by when, and what to produce. The judgment criteria field describes that criteria to judge that “output of the process is good.”

JKK's business processes can also be seen as business processes. JKK clarifies the completeness conditions for each business process element. The requirement organizing sheet of JKK hasn't exceptional output that is the key item of the proposed approach in this paper.

FRAM (Functional Resonance Analysis Method) [13] has been used to analyze complex functional resonances of socio-technical systems through functional networks. The FRAM function is defined by hexagonal nodes with six sides. These sides are corresponded to six aspects which are Input (I), Output (O), Time (T), Control (C), Resource (R), and Precondition (P). The output side of a function can be connected to the other five sides of other functions. FRAM provides useful means for safety analysis. Possible aspect relationships are $\langle O, I \rangle$, $\langle O, T \rangle$, $\langle O, C \rangle$, $\langle O, R \rangle$, and $\langle O, P \rangle$. Here, $\langle X, Y \rangle$ is where X and Y are functional aspects. Although FRAM aspects had been used to analyze irregular variations for safety and security, the integration with NFR isn't clear.

2.7. Assurance case

Security and Safety requirements of systems are confirmed by assurance cases. The safety case, the assurance case, and the dependability case are currently the focus of considerable attention for the purposes of assuring that systems are safe. Methods have thus been proposed for representing these cases using Goal Structuring Notation (GSN) [14–16]. In the absence of any clearly organized guidelines concerning the approach to be taken in decomposing claims using strategies and the decomposition sequence, engineers have often not known how to develop their arguments.

Yamamoto [17] proposed a method to introduce quantitative attributes for GSN is proposed to evaluate architectures based on quality claims, such as security and safety. The attribute propagation rule from evidence attributes to the top claim attribute is also described. The method solves quantitative evaluation of argument issues for assuring security, and safety architectures.

Patu and Yamamoto [18, 19] proposed dependability cases with focus on security. Test result has been used as evidence of dependability argument mitigates security threats.

Mohamad, Steghofer, and Scandariato [20] surveyed published papers on Security Assurance Case (SAC). They concluded that there are numerous papers discussing the importance of SAC and their usage scenarios, the literature is still immature with respect to concrete support for practitioners on how to build and maintain a SAC.

Yamamoto, Morisaki and Atsumi [21] proposed a method named SPRME for creating assurance cases based on the following five items. The SPRME approach is used to develop the FNRME proposed in this paper.

[Subject] structure of the software to be guaranteed

[Property] expected quality characteristics

[Risk] assumed risks

[Measure] countermeasures against for risk

[Evidence] evidence of the results of the countermeasures.

3. Non-Functional Requirements Extraction from Exceptions

3.1. Relationship between NFR and exceptions

Of Robertson and Robertson's eight types of non-functional requirements, regarding appearance requirements, usage requirements, organizational culture requirements, and legal system requirements, in order to determine whether a system satisfies these requirements, it is necessary to A model of these requirements is defined and the behaviour of the system is determined. We need to ensure that we do not deviate from the model. These non-functional requirements can be described by detecting deviations from the model as exceptions.

Regarding maintainability requirements, by checking the status of system components, it is necessary to detect conditions that cause system abnormalities as exceptions, and to notify replacement of components.

Regarding performance requirements, by monitoring whether the input is processable, inputs that cause performance abnormalities can be detected as exceptions. For example, you can restrict input that exceeds the range based on the upper limit of acceptable data.

Regarding operational requirements, deviations from the normal state can be detected as exceptions by managing state transitions based on system operation.

Regarding security requirements, security threats can be detected as deviations by checking the input content. For example, since SQL injection requires inputting a long string, input exceeding the upper limit of the input string can be detected as an exception. Furthermore, it is possible to describe the realization of security requirements by defining a function that detects CIA (Confidentiality, Integrity, and Availability) exceptions, which are the three aspects of security.

In this way, non-functional requirements can be described by detecting threats to functional requirements as exceptions and describing functions as countermeasures against the threats.

Additionally, when the system detects the above exception, it needs a function to warn of deviations from non-functional requirements in order to notify the outside world of the exception. Here, if no exception occurs, the non-functional request has been fulfilled.

3.2. Extraction method of NFR from exceptions

The method of extracting non-functional requirements mentioned above is summarized as follows.

[Input] Functional requirements set: F, non-functional requirements set: N

[Output] Risk set: R, countermeasure set: M, exception set: E

[Method]

For every non-functional requirements N, do the following:

Identify the associated functional risk R.

Describe the measures to eliminate or mitigate the functional risk R.

Describe the exception E of risk countermeasure M.

Describe the external notification function of exception E.

(end of method)

As you noticed that the method borrowed the fundamental concept from SPRME mentioned above. The F, N, R, M, and E correspond to S, P, R, M, and E of SPRME, respectively. The metamodel contains of the above method elements is shown in Fig.1. The metamodel can be called as FNRME.

FNRME replaced S, P, and E of SPRME by Functional Requirements, Non-Functional Requirements and Exception.

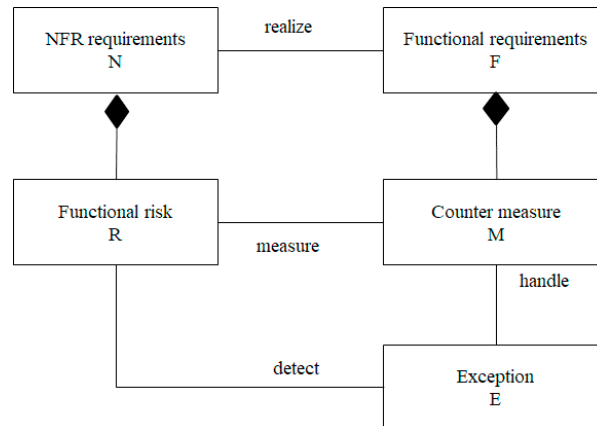


Fig. 1. Metamodel of NFR extraction method.

4. Comparison of NFR approaches

The following Table shows the result of comparison among NFR framework, Control case and HPM by using FNRME. The table also compares external entity, business process, integrity review means and representation of those approaches. In case of NFR framework, external entity and business process are described by i* framework [6]. Although the integrity review of NFR framework is checking operationalization of soft goals to operational goals, it lacks integrity rules. Control case template provides means to check integrity of FNRME constituents. HPM can check integrity of timing by state transition table.

Table1. Comparison of NFR framework, Control case and HPM

Approaches	NFR framework	Control case	HPM
Function	FR	Use case	DFD
Non-Functional requirements	NFR	NFR	Realtime specification
Risk	Soft goal	Threat	Unexpected Timing
Measure	Operational goal	Operation condition	Process of DFD and CFD
Exception	Exceptional operation	Vulnerability	exceptional event
External entity	Actor (i* framework)	Threat source	Externals
Business process	Task (i* framework)	Business constraints	External process
Integrity review means	Operationalization	Control case template	State table
Representation	Goal diagram	Use case extension	CFD

5. Case study

The following is the requirements specification of insulin pump control software [22].

- (R1) Insulin pump control software uses sensors implanted in the patient to measure blood parameters proportional to blood glucose levels.
 (R2) Blood parameters are sent to the pump controller.
 (R3) The pump controller calculates the sugar level and amount of insulin needed.
 (R4) The pump controller sends a signal to a small pump to administer insulin through a needle implanted in the patient.
 (R5) The insulin pump delivers 1 unit of insulin in response to a 1unit pulse from the pump controller.

NFR framework, Control case, and HPM are compared by extracting maintainability requirements by describing exceptions on equipment of the insulin pump system. In this insulin pump system, we consider blood sugar sensor and insulin tank as equipment.

For blood sugar sensor, sensor malfunction may be occurred as unexpected exception.

For insulin tank, insulin capacity may be shorted as insulin pump continuously injects insulin for patient.

Therefore, it is necessary to prevent dangerous accident by detecting these exception and replacements of equipment.

We demonstrate through the case studies that it is possible to relate non-functional requirements and functional exceptions, which was unclear using conventional methods. Non-functional requirements and exceptions of the three approaches are studied below. The proposed method is used to clarify and connect NFRs, risks, countermeasures and exceptions.

5.1. NFR framework

Fig.2 showed the relationship between maintainability and the maintain equipment functional requirement. Operation confirmation is an operation soft-goal which correspond to status check of equipment. The check result is notified by status notification operational soft-goal. If an exception of sensor status is detected, then change request of equipment is issued.

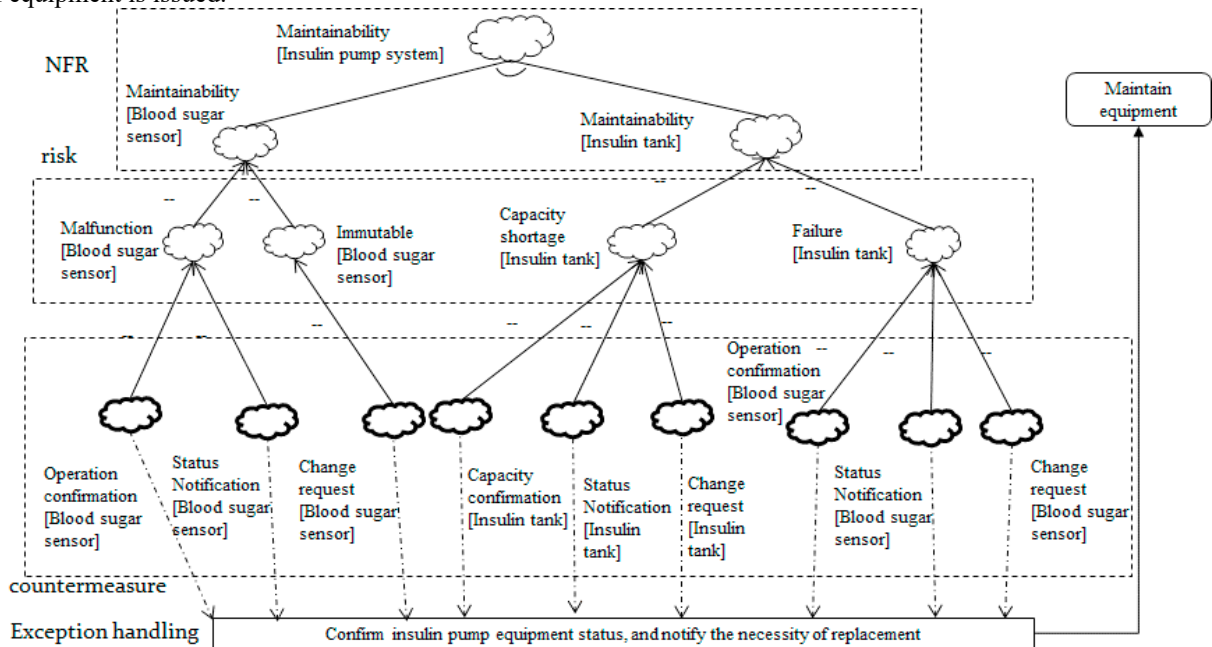


Fig. 2. Maintainability and functional requirements of insulin pump system in NFR framework.

In Fig.2, the maintainability is divided into the maintainability of blood sugar sensor and insulin tank. Then the risks of these two divided maintainability are described as shown in dotted risk rectangle. For these risks,

countermeasures are described by the functional goals pairs consists of confirmation, status notification, and change request.

5.2. Control case

Table 2 shows the control case of the insulin pump maintainability requirements. In Table 2, functions and exceptions are described as associated use case and vulnerability, respectively. The functional exception is explicitly described as the “notify irregular status” uses case. However, this kind of exceptional uses case description manner is unknown in the original control case approach.

Table 2. Control case of Maintainability of insulin pump system.

Items	Definition
Operating condition	Maintain equipment
Description	The system’s response of insulin injection may become unsafe for patients, in sake of uncertain sensor status and shortage of insulin tank capacity. This control case defines the least sensing interval of blood sugar and the least insulin tank capacity that system must meet in order to mitigate the risk.
NFR category	Maintainability
Associated Use Cases	Confirm sugar blood sensor and insulin tank capacity, notify irregular status, change request for equipment
Business constraints	Blood sugar needs to be sensed. Insulin needs to be provided
Vulnerability	unstable status in blood sensor, insulin tank capacity shortage
Threat source	Blood sugar sensor, insulin tank capacity
Operation condition	1.Sensing interval of blood sugar 2.Least insulin tank capacity of injection

5.3. HPM

Fig.3 shows CFD of HPM for insulin pump control system. Dotted arrows show control flows for exceptions. The thick short black line represents the existence of state transitions to control exceptional events. Circles and rectangles are processes and external entities, respectively.

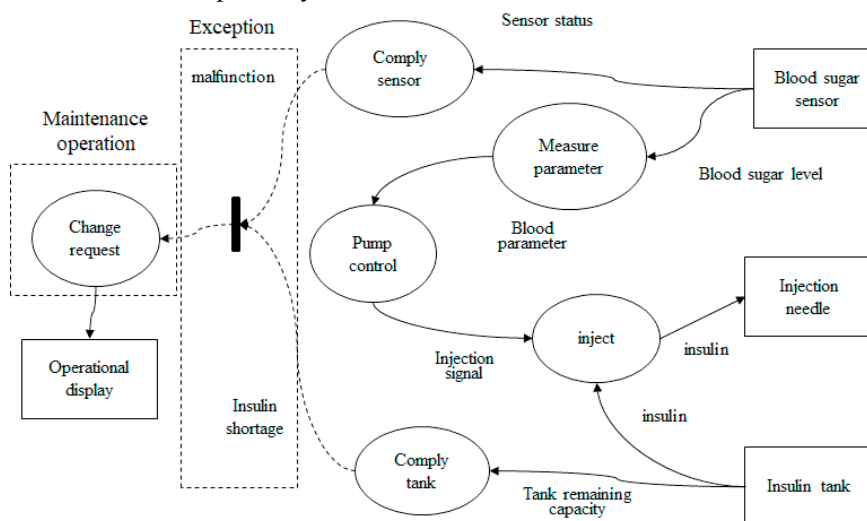


Fig. 3. Maintainability and functional requirements of insulin pump system in HPM.

Although HPM is able to describe exceptional events as control flows from functions as shown in Fig.3, it is impossible to describe non-functional requirements explicitly in HPM. Instead, the maintenance operation is described as the corresponding countermeasure of the exceptional event as shown in dotted rectangle named maintenance operation.

6. Discussion

This section discusses the novelty, effectiveness, and limitations of the proposed approach.

6.1. Novelty

As there hasn't been relationship analysis between non-functional requirements and exceptions, the approach to realize non-functional requirements with exceptions was unclear. Moreover, there hasn't been a common method for comparing non-functional and functional requirements integration approaches comprehensively. We proposed FNRME model to describe the relationship between non-functional requirements and exceptions, as shown in Fig.1. By using FNRME, it's possible to compare the approaches of NFR framework, control case and HPM. The case study also clarifies that these previous methods can be extended to integrate NFR and FR.

6.2. Effectiveness

The examples in section 5 shows the applicability of the proposed method to define non-functional requirements of insulin pump control system in NFR framework, control case and HPM. Furthermore, as the three methods commonly compared using FNRME model in Table 1, the seamless integration possibility of these different methods has been clear. This demonstrates flexibility in comparing non-functional requirements methods while describing each respective NFRs and exceptions. It's also noteworthy that the FNRME constituent elements serve as common factors for extracting NFRs. NFRs in information systems can be realized from these common constituent elements.

6.3. Identification of exceptions

The point of the paper is how do we can confirm the NFR. We can assure the NFR is satisfied by the evidence that we detect no exception related to the NFR. If valid conditions of entities are identified, it is easy to identify exceptions of them. If functional requirements are identified, its input and output are also defined. If expected inputs and outputs are clearly defined, the exceptions are easily defined as un-expected inputs and outputs. To think identifying exceptions is difficult is that it is difficult to define valid input and output conditions. If valid condition is not to define, it is impossible to show the correctness of the functional requirements.

There are existing requirements specification approaches that identify exceptional requirements. For example, Shui et al [23] proposed the exceptional use case template that clarifies exceptional conditions of functional requirements. These approaches demonstrate the importance of clarifying exceptional requirements.

6.4. Limitations

Since our case study in this paper focused only on maintenance exceptions of insulin pump control system, it is evident that investigating other NFRs and systems is necessary to develop a comprehensive model of NFRs from exceptions. Furthermore, as the approaches in Table 1 only compares three approaches, there is a need to evaluate the generality of FNRME model of NFR extraction from exceptions. Additionally, we need to examine other NFRs other than maintainability. Also, there is a need for quantitative evaluation of the proposed approach.

7. Summary

In this paper, we proposed a FNRME model for defining NFRs through exceptions. As a result, we clarified the following.

- (1) FNRME can express the relationship between NFRs and FRs through exceptions
- (2) FNRME can compare different NFR approaches
- (3) It was clarified that FNRME has the potential to integrate different NFR approaches.

In the future, it is necessary to develop a method that integrates NFR approaches, as well as advancing the evaluation that quantitatively clarifies the effectiveness of this method.

In addition, although the case study only targets maintainability requirements for one embedded system. For example, it is necessary to evaluate applicability to the other Non-functional requirements such as usability and compliance.

References

- [1] Robertson, S. and Robertson, J. (2012) "Mastering the Requirements Process: Getting Requirements Right." *Addison-Wesley Professional*.
- [2] Navarro, I., Leveson, N., Lunqvist, K. (2010) "Semantic decoupling: reducing the impact of requirements change." *Requirements Engineering*, Springer, 15(4):419-437
- [3] Leveson, N. (1995) "SAFWARE – System Safety and Computers," Addison Wesley.
- [4] Moreira, A., Brito, I., and Araújo, J. (2002) "Crosscutting quality attributes for requirements engineering." 14th ICSE and SEKE'02, *ACM Press*: 167-174.
- [5] Losavio, F. (2014) "Unified Process for Domain Analysis integrating Quality, Aspects and Goals." *CLEI Electrical Journal*: 17(2) Montevideo ago.
- [6] Chung, L., Nixon, B., Yu, E., Mylopoulos, J. (2000) "Non-Functional Requirements In Software Engineering." *Kluwer Academic Publishers*.
- [7] Subramanian, N. and Zalewski, J. (2014) "Quantitative Assessment of Safety and Security of System Architecture for Cyberphysical Systems Using the NFR Approach." *Systems Journal, IEEE*: (99) 1-13.
- [8] Zou, J. and Pavlovski, J. (2006) "Modeling Architectural Non Functional Requirements: From Use Case to Control Case," *ICEBE '06*: 1-8.
- [9] Pavlovski, J. and Zou, J. (2008) "Non-Functional Requirements in Business Process Modeling." *5th Asia-Pacific Conference on Conceptual Modelling*: 103-112.
- [10] Hatley, J. and Pirbhai, A. (1987) "Strategies for Real-Time System Specification." *Dorset House Publishing*
- [11] Grigori, D., Casati, F., Dayal, U., Shan, M. (2001) "Improving Business Process Quality through Exception Understanding, Prediction, and Prevention." *27th VLDB Conference*.
- [12] Sasaki, S. (2014) "Self-process completion - Quality is built in the process." *JSQC selection, Japan Society for Quality Control*. (in Japanese)
- [13] Hollnagel, E. (2012) "FRAM - the Functional Resonance Analysis Method: Modelling Complex Socio-Technical Systems." *Boca Raton: CRC Press*.
- [14] Kelly, T. (1997) "A Six-Step Method for the Development of Goal Structures." *York Software Engineering*.
- [15] Kelly, T. (1998) "Arguing Safety, a Systematic Approach to Managing Safety Cases." PhD Thesis, Department of Computer Science, University of York.
- [16] Kelly, T., Weaver, R. (2004) "The Goal Structuring Notation – A Safety Argument Notation, Proceedings of the Dependable Systems and Networks." *2004 Workshop on Assurance Cases*.
- [17] Yamamoto, S. (2015) "Assuring security through Attribute GSN." *5th International conference on IT convergence and security (ICITCS)*. *IEEE*: 1–5
- [18] Patu V., Yamamoto S. (2013) "How to develop security case by combining real life security experiences (evidence) with d-case. *Procedia Comput Sci* 22:954–959.
- [19] Patu, V., Yamamoto, S. (2013) "Identifying and implementing security patterns for a dependable security case—from security patterns to d-case." *16th international conference on computational science and engineering*: 138–142.
- [20] Mohamad, M., Steghofer, J., Scandariato, R. (2021) "Security assurance cases—state of the art of an emerging approach." *Empirical Software Engineering*: 26(70) 1-43.
- [21] Yamamoto, S., Morisaki, S., and Atumi, N. (2017) "A unified approach on assurance case development method based on models." *JSAI Technical Report*, 2015(KSN-017): 1-6. (in Japanese)
- [22] Sommerville, I. (2010) "Software Engineering." *Pearson Education*.
- [23] Shui, A., Mustafiz, S., Kienzle, J., Dony, C. (2005) "Exceptional Use Cases." *MoDELS 2005, Lecture Notes in Computer Science*, 3713:568-583.