A Proposal of Descriptive Pattern for Maintainability Requirements

Yuki Sanomachi
Shibaura Institute of Technology
Tokyo, Japan
e-mail: ma18051@shibaura-it.ac.jp

Tsuyoshi Nakajima
Shibaura Institute of Technology
Tokyo, Japan
e-mail: tsnaka@shibaura-it.ac.jp

***Abstract*—To clearly define quality requirements is crucially important for developing high quality systems and software. Unlike usability and security requirements, maintainability requirements often come from developers themselves. Therefore, their descriptive patterns have not been discussed so much in spite of their importance. This paper proposes a descriptive pattern for maintainability requirements based on the quality requirements framework in the ISO/IEC 25030:2019. The proposed descriptive pattern covers maintainability requirements using all the measures in ISO/IEC 25023 and enables machine checking their correctness and unambiguity.**

Keywords-quality requirements; maintainability; SQuaRE; standardization.

I. INTRODUCTION

Information and Communication Technology (ICT) systems have been used in various places and situations, and therefore, their failures can have a large impact on the society. Therefore, it is required not only to fit them to various needs and usage scenes, but also to ensure their quality through careful consideration on social impact [1].

Development of a quality ICT system is required to meet its quality requirements as well as its functional ones. Quality requirements cover many views of the target system, among which quality views related to system behavior, such as usability and Security. These have been discussed thoroughly in separate communities to standardize manners to write their quality requirements.

In contrast, quality views related to the internal structure of the system, such as maintainability and portability, are not properly discussed with respect to standardization and the manner in which they are specified.

In the systems and software engineering field, the reality is that quality requirements are specified in a variety of manners, most of which are not properly written without being separated from functional requirements [2].

ISO/IEC 25000 series (SQuaRE: Systems and software Quality Requirements and Evaluation) are developed to provide a framework for quality definition and evaluation, including quality models and measures, which cover an exhaustive set of quality views. In addition, ISO/IEC 25030: 2019 [3] in the SQuaRE series provides a framework for defining quality requirements using the quality models and measures.

In this paper, we propose a descriptive pattern for specifying maintainability requirements based on the specification format defined in ISO/IEC 25030:2019. The proposed descriptive pattern covers various kinds of maintainability requirements and enables machine checking their correctness and unambiguity.

In this paper, Section II describes the related work, and then, Section III proposes a descriptive pattern for maintainability. Section IV gives a qualitative evaluation of the proposed pattern. Section V summarizes this paper and gives future work.

II. RELATED WORK

ISO/IEC 25010 [5] defines the product quality model with eight quality characteristics. Maintainability is one of them, which provides five sub characteristics: modularity, reusability, analyzability, modifiability, and testability, whose definitions are shown in TABLE 1.

TABLE I DEFINITIONS OF SUB－CHARACTERISTICS OF MAINTAINABILITY [5]

Sub-characteristic	Definition
modularity	degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components
reusability	degree to which an asset can be used in more than one system, or in building other assets
analyzability	degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified
modifiability	degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality
testability	degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met

ISO/IEC 25023 [6] product quality measures provides 86 quality measures corresponding to quality sub characteristics, including 13 measures for maintainability characteristic. Japan Users Association of Information Systems (JUAS) published a guideline which defines their own quality measures other than those of ISO/IEC 25023 [4]. These give

 ISBN: 978-1-61208-752-8

recommendation on what measures should be used for maintenance requirements. However, these do not guide how to define maintenance requirements themselves. Therefore, there are needs for descriptive patterns to guide it.

ISO / IEC 25030: 2019 [3] provides the following specification format for specifying quality requirements.

- **Target entity:** *Components of the system*
- **Selected characteristic:** *modularity*
- **Quality goal with conditions:** *Reducing the coupling between any two of components*
- **Quality measure:** *(MMo-1-G) coupling of components*
- **Target value:** *1*
- **Acceptable range of values:** *0,98 – 1,00*

It is recommended to use this format for specifying not only the scope and goal of the quality requirement, but also quality measures with its target value and acceptable range of the value.

III. DESCRIPTIVE PATTERN FOR SPECIFYING MAINTAINABILITY REQUIREMENTS

In this paper, we propose a descriptive pattern for maintainability requirements with the aim of standardizing the manner to specify them. We set the following four requirements for the descriptive pattern:

(1) The descriptive pattern for specifying maintainability requirements shall cover all types of maintainability requirements with as few descriptive patterns as possible.
(2) Maintainability requirement statements conformed to the pattern shall be natural to readers.
(3) Maintainability requirement statements conformed to the pattern shall prevent from ambiguities.
(4) The pattern shall enable machine checking correctness and unambiguity for maintainability requirement statements.

In order to create the descriptive pattern of the maintainability requirements, we did the following analysis:

i. Create a table of maintainability measures merging those from the ISO/IEC 25023 and JUAS guidelines.
ii. Extract what achievement to be measured in the corresponding quality sub characteristic as the "quality goal".
iii. Parameterize the quality measure so that it 2can be limited to the appropriate level of application. The parameters include:
    - Scope of the target entity
    - Criteria for the evaluation
    - Context for application (evaluation period, subjects, etc.)
iv. Create descriptive patterns and try using them to write requirement statements.

As a result of the above analysis, we propose the following descriptive pattern for maintainability requirements.

In order to *Quality goal*, *quality measure* shall be [greater | smaller] than *Target value*.

Quality goal = [improve | increase | suppress | decrease] *Attribute* of *Target entity*] | *Outcome of use*

The two rows of *Quality goal* and *Quality requirement statement* are added to the original table of quality measures to be TABLE 2. In TABLE 2, for example, if *Quality goal* is "increase the independency of system components," *Quality measure* is "the coupling of components," and *Target value* is "99.0%"," the quality requirement statement goes to:

In order to increase the independence of system components, the coupling of components shall be greater than 99.0 %.

IV. EVALUATION

All the maintainability requirements in TABLE 2 can be specified naturally using the proposed descriptive pattern. This proves to meet the requirements (1) and (2).

Quality requirement statements conformed to the proposed descriptive pattern have all the items of the specification format in ISO/IEC 25030:2019, which is designed to prevent the quality requirement statements from being ambiguous, which meets the requirement (3).

SE Suite [7] is a tool for describing, checking, and evaluating quality in requirement specifications, and consists of the following three tools:

- Requirements Quality Analyzer (RQA),
- Requirements Authoring Tool (RAT)
- Knowledge Manager (KM)

RQA provides a checking function for general requirement statements with designated descriptive patterns. We can use this tool to implement machine checking maintainability requirements using the proposed description pattern and vocabularies defined in KM. SE Suite can check the following points:

- whether the statement is syntactically correct or not
- whether terms are defined or not
- whether relationship between terms are appropriate or not
- whether the context of application of the quality measure is appropriate or not
- whether the range of the quality measure is appropriate or not

 ISBN: 978-1-61208-752-8

Figure 1 shows an example of checking layers of SE Suite for the maintainability requirement statement using the proposed pattern.

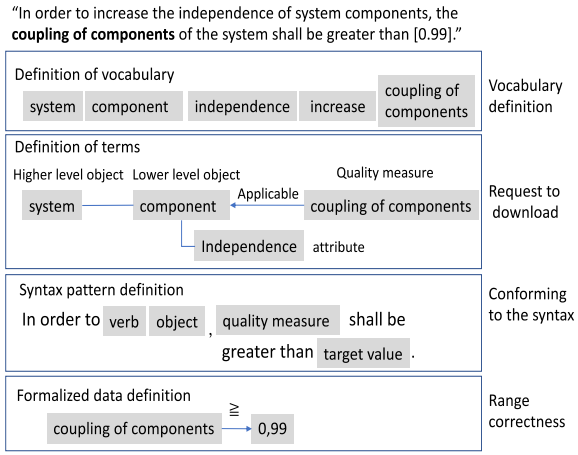

Figure I Checking layers of SE Suite for the maintainability requirement statement using the proposed description pattern

By implementing the description pattern proposed with SE suite, it is mechanically possible to detect errors, omissions, and ambiguities in the requested sentences.

This shows that the proposed description pattern meets requirements (4).

V. CONCLUSION AND FUTURE TASKS

In this paper, we proposed a descriptive pattern for maintainability requirements, which proves to cover various types of them. Using SE Suite, this pattern enables a variety of checking for maintainability requirement statements.

The proposed descriptive pattern can be applied to all the maintainability measures defined in ISO/IEC 25023 and JUAS, but it may not be limited to them. On the contrary, the single pattern might not be enough to specify some other maintainability requirement statements. Therefore, we should consider quantifying quality attributes, such as target question metrics (GQM). The proposed pattern should be examined to write more examples of maintainability requirements, which strengthen our belief on the proposed pattern to meet the requirements for it.

ACKNOWLEDGMENT

This research was conducted as a project of "International standardization for quality model and evaluation of system and software (Japan Standards Association)" as a part of the industrial standardization promotion business of Ministry of Economy, Trade and Industry, FY2019.

REFERENCE

[1] Information-technology Promotion Agency Japan, " Information system failure status", 2018, pp. 1–3.

[2] J. Eckhardt, A.Vogelsang, and D. Méndez Fernández, "Are non-functional requirements really non-functional? an investigation of non-functional requirements in practice", Proc. International Conference on Software Engineering(ICE16), ICE Press, Nov. 2016, PP. 832-842, doi:10.1145/2884781.2884788.

[3] ISO/IEC Software Engineering, ISO/IEC25030 Software Product Quality Requirements and Evaluation (SQuaRE) Quality requirements framework, 2019.

[4] JUAS ed, "A guideline for specification of non-functional requirements (UVC project II)", 2008, pp. 89–100.

[5] ISO/IEC Software Engineering, ISO/IEC25010 Software Product Quality Requirements and Evaluation (SQuaRE) System and software quality models, 2011.

[6] ISO/IEC Software Engineering, ISO/IEC25023 Software Product Quality Requirements and Evaluation (SQuaRE) Measurement of External Quality, 2016.

[7] The REUSE Company.Systems Engineering Suite, https://www.reusecompany.com/systems-engineering-suite, (accessed 2019-11-17).

TABLE Ⅱ PROPOSED DESCRIPTIVE PATTERN FOR MAINTAINABILITY REQUIREMENTS AND ITS APPLICATION TO QUALITY MEASURES

Quality sub-characteristic	Quality goal	Quality measure	Measurement function	Quality requirement statement
Modularity	Increase the independence of system components.	Coupling of components	X=A/B A=Number of components which are implemented with no impact on others B = Number of specified components which are required to be independent	In order to increase the independence of system components, the coupling of components of the system shall be greater than [target value].
	Increase module cohesion	Cyclomatic complexity adequacy	X = 1– A/B A = Number of software modules which have a cyclomatic complexity score that exceeds the specified threshold B = Number of software modules implemented	In order to increase module cohesion, the cyclomatic complexity adequacy should be greater than [target value].
Reusability	Increase the number of reusable components (or modules).	Reusability of assets	X = A/B A = Number of assets which are designed and implemented to be reusable B = Number of assets in a system	In order to increase the number of reusable components (or modules), the reusability of assets (applicable range, criteria for being an asset, Criteria for reusability) shall be greater than [target value].
	Improve coding quality of modules	Coding rules conformity	X = A/B A = Number of software modules conforming to coding rules for a specific system B = Number of software modules implemented	In order to improve coding quality of modules, the coding rules conformity (scope, coding code) shall be greater than [target value].
Analyzability	Increase the adequacy of data used to trace causes of the system failures	System log completeness	X=A/B A = Number of logs that are actually recorded in the system B = Number of logs for which audit trails are required during operation	In order to increase the adequacy of data used to trace causes of the system failures, the system log completeness shall be greater than [target value].
	Improve the accuracy and efficiency of identifying causes of the system failure.	Diagnosis function effectiveness	X=A/B A = Number of diagnostic functions useful for causal analysis B = Number of diagnostic functions implemented	In order to improve the accuracy and efficiency of identifying causes of the system failure, the diagnosis function effectiveness shall be smaller than [target value].
		Diagnosis function sufficiency	X=A/B A = Number of diagnostic functions implemented B = Number of diagnostic functions required	In order to improve the accuracy and the efficiency of identifying causes of the system failure, the diagnosis function sufficiency shall be greater than [target value].
	Improve the readability of the program.	Program source comment rate	X=A/B A = Implemented comment rate B = Comment rate defined by the organization	To improve the readability of the program, the program source comment rate (application range) shall be more than [target value].
Modifiability	Improve the accuracy and efficiency of system correction.	Modification correctness	X = 1 – (A/B) A = Number of modifications that caused an incident or failure within a defined period after being implemented B = Number of modifications implemented	In order to increase the accuracy and efficiency of system correction, make modification correctness (target period) less than [target value].
		Modification capability	X=A/B A = Number of items actually modified within a specified duration B = Number of items required to be modified	In order to improve the accuracy and efficiency of system correction, the modification capability shall be smaller than [target value].
	Improve the appropriateness of system modification management.	Change Content Documenting Rate	X=A/B A = Number of features documented and subject to review B = Number of functions with program change	In order to improve the appropriateness of system modification management, the change content documentation rate shall be greater than [target value].
Testability	Increase the sufficiency of functions to support test execution.	Test function completeness	X=A/B A = Number of test functions implemented as specified B = Number of test functions required	In order to increase the sufficiency of the function to support test execution, the test function completeness shall be greater than [target value].
	Increase the possibility of independent testing.	Autonomous testability	X=A/B A = Number of tests that can be simulated by stub among the tests which depend on other systems B = Number of tests which depend on other systems	In order to increase the possibility of independent testing, make autonomous testability more than [target value].

 ISBN: 978-1-61208-752-8