International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

Literature Review on Extended Use case in Modeling Non-functional Requirement

Nina Fadilah Najwa*[1], Muhammad Ariful Furqon[2], Faizal Mahananto[3]

*[123] Department of Information System, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
nina.fadilah.najwa16@mhs.is.its.ac.id[1], ariful.furqon16@mhs.is.its.ac.id[2], faizal@is.its.ac.id[3]

ABSTRACT

Complexity in defining non-functional requirements happened because it has many aspects. It is very difficult for stakeholders to know non-functional requirements in detail which are actually needed and has an impact on the abandonment of non-functional requirements elicitation. So it is required to find out the importance of modeling of non-functional requirements by using extended use case. In order to review the topic, research questions are formulated then defining keywords for finding a literature from the resources. The result of finding literature shows that the use of extended use case can ease the modeling of non-functional requirements because of the ease of the annotation and more familiar to use. There are eight steps for modeling non-functional requirements with an extended use case. Furthermore, the extended use case can model the various categories of non-functional requirements such as security, availability, performance, robustness, visibility, analyzability, manageability, completeness, correctness, redundancy, accessibility and adaptivity.

Keywords: Extended Use Case, Non-Functional, Requirement Software Engineering

I. INTRODUCTION

Elicitation is one of the processes that are extremely important in the software development and part of techniques for requirement analysis. In fulfilling software requirements, stakeholders need to understand the requirement for achieving the objectives and benefits of the system to be made. Evolving requirements categorized into functional requirements and non-functional requirements [1]. Functional requirements are statements from system service which is must be available in that system, such as how the system interaction with the type of input data and how the system should act in particular situation [2]. Besides, non-functional requirements are one of the keys criteria to distinguish between many types of system in the software development. In general, non-functional requirements in software development specified in addition to the requirement, as performance, reliability, security, accuracy, etc., which is in its early stages software development seems similar with functional requirement [1].

Requirement elicitation has long been focusing on one side that is functional requirements namely objects and about functional task activity in the system [3]. Neglect gainstof non-functional requirements will have an impact on failure in the software development. In general, problems that often happens is very lack of stakeholder's attention for the understanding the non-functional requirements of the system [1]. Non-functional requirements are too subjective to clearly specificated because of the impacts from system analyst and stakeholder [4]. This case also because complexity in defining non-functional requirements have many aspects, and thus it is very difficult for stakeholders to find out by details of non-functional requirements required actually.

According to the research of Cysneiros and Yu still at least technique in eliciting for modeling non-functional requirements. Some approach that integrates non-functional requirements will take the expense of maintaining costs [1]. This obviously different to elicitation technique in modeling functional requirements that have had lots of technique.

Sometimes, functional requirements and non-functional requirements equated meaning and give rise to ambiguity. Since the non-functional requirements connected to the functional requirements, this creates conflicts between stakeholders and developers. For example, the security system can use the keyword level two or biometric system but will be increased production costs that deal with non-functional requirement system [1]. Thus, non-functional requirements not be likened to functional requirements in software development.

Generally, in modeling software requirements, tools used in Unified Modelling Language (UML) which classifies different requirements and visualize into the design system [4]. Among types of the diagram in UML, the use case is the most often used in described functional requirements system [5]. But, the research has proved by use case can be described non-functional requirements but having many irregularities and too much ambiguity if combined with functional requirements. This is because of use case is not the only model for the system, but also the model for the human action.

On modeling use case in Unified Modelling Language (UML), there is an element the relationships, which represented whether a use case depends on use case another or is an extension of use case another. Hernandez [6] classify relationship to include and extend. Include is a relationship indicating that functionality of a use case depends on use case another. While extend is a relationship in use case where done the addition of additional requirements especially non-functional requirements defined in an extended use case. According to Turner [5] to model human interaction, system, and data requires a modeling use case extended by using relationship 'extend'. Modeling performed with an extended use case are usually related to the requirements of performance system or non- functional requirements software that other. So, that in the process modeling software requirements to defines and model non-functional requirements with better to use extended use case diagram to avoid development software that poor quality and unstable.

Based on the research has been conducted by previous research associated with modeling non-functional requirements, then modeling extended use case suitable for model non-functional requirements software. The extended use case can be used to elicit requirements such as presented by Rahman [1] that non-functional requirements elicitation is one of essential prioritized from the perspective of stakeholders. Besides used to elicit and model non-functional requirements what it must present on the system, according to Zou [7] extended use case also can help the developer to understand what the system needs for non-functional requirements. In addition problems complexity use case probably one of the things that should be avoided in modeling non-functional requirements with the extended use case. According to Kaur [4], relation or relationship between use case will add complexity in use case. So, it is important to heed the complexity of a use case in defining non-functional requirements in use case.

In writing this review literature, writers will review issues concerning on modeling non-functional requirements software using extended use case as a technique to model non-functional requirements. The purpose of this literature review should assess some paper associated with non-functional requirements and modeling extended use case. Based on the study of the research, writers generalize intents and purposes of paper associated with modeling non-functional requirements and concluded research results papers beforehand so that is expected to be more understandable. In addition, the contribution given by this literature review is supported or corroborate the results of studies that have been done before relating to modeling non-functional requirements by using extended use case. Strengthening the results of the study formerly given by writers of critical thinking stated at a discussion based on previous studies. It is expected that literature review it is not just being the summary of the previous studies but also can trigger researchers another to do research on the topic of modelling non-functional requirements with the extended use case. The last result expected that is, to know the advantages and the importance of extended use case in model non-functional requirements, it also needs to pay attention to the weakness of modeling is problems complexity that may be caused by modeling extended use case.

II. RELATED WORKS

A. Non-functional Requirement

Zou et al describe non-functional requirements as an important limitation that must exist during software

development [7]. Non-functional requirements should be defined as early as possible otherwise, they will cause problems related to the quality of the developed software. According to Casamayor, et al [8] the examples of non-functional requirements are security, performance, availability, extensibility, and portability. However, according to ISO 9126 described by Zou, et al [7] There are six categories of non-functional requirements including maintainability, functionality, portability, efficiency, usability, and reliability. According to Zhang [9], Non-functional requirements are essential for success in software development considering that in software development there is a constraint which must be defined so that the quality of the developed software can be as expected by the stakeholders.

On the other hand, according to Olmsted [10] in fact, in the software development, nonfunctional requirements are treated like secondary requirements and are often ignored until the end of the software development. Some studies suggest that non-functional requirements are difficult to model, develop, or test so that these non-functional requirements are often overlooked. According to Singh, et al [11] as a result of underestimating non-functional requirements on software development may lead to the failure of the software development project. A similar opinion was also presented by Mahmoud [12] that failure to identify non-functional requirements could have an impact on the quality of the software developed. More specifically described that non-functional requirements tend to be connected through a variety of interdependencies that are interconnected with functional needs. So if non-functional requirements are not defined in the early stages of software development, it will impact the next stage of software development.

According to Liu, et al [13] in recent years, there has been an increasing awareness of the importance of non-functional requirements for developing a user satisfied software. Several studies related to non-functional requirements have been made for the purpose of providing a systematic way to address non-functional requirements in the early stages of software development. Several approaches have been made to model non-functional requirements in the early stages of software development.

According to Chopra [14], non-functional requirements have been considered as unique requirements which not found in functional requirements and must remain defined and modeled on software development.

B. Extended Use Case

The purpose of use case modeling is to model the requirements of people without high training or special skills to understand and describe them [15]. In a use case diagram, it consists of four elements that work on a system: the system itself, the actor that interacts with the system, the service (use case) fulfilled by the system, and the relationships between these elements [16]. The relationship on UML (include, extend, generalization) describes the structural view of the requirements. Extend is an event of a use case that may be incremented by the addition of a behavior defined in an extending use case [6]. Two use cases connected to a «extends» relationship if one use case (known as a base use case) is implicitly a behavior of one use case (called an extension use case) in a specific location. Extension use cases are an execution only when special conditions are met in basic use cases [16]. There are several reasons for submission to the literature using the «extends» relationship in the use case model.

III. METHODOLOGY

In this phase will be described literature methodology which is based on guidelines of Kitchenham [17]. But the basis of reviews literature do is to collect and evaluate the research that deals with questions raised issues, then produce ambiguity, and capability [18]. As for the purpose of literature review according to Kitchenham, is to identify, evaluate, and research available relating to questions to research or topics, or phenomenon of being interesting [17].

The methodology that used in literature review refers to the methodology described by Kitchenham [17, 19], which includes several phases, they are:

1. Defining research question
2. Literature sources
3. Determining the keywords of literature search
4. Literature selection
5. Extracting data and synthesis

A. Research Question

This is literature review specified based on research question:

1. RQ 1: What is the problem raised by previous research regarding the extended use case in non-functional requirements modeling?
2. RQ 2: Why is an extended use case important to use in non-functional requirements modeling?
3. RQ 3: How to use extended use case to model non-functional requirements?

B. Literature Sources

A source database which is used to search literature be limited to the international journal sites to obtain the journal with good quality, namely:

1. IEEE
2. Science Direct
3. ACM.

C. Keywords of Literature Search

Following keywords used to make search literature associated includes some keywords. They are:

1. Extended Use Case
2. Non-functional Requirement
3. Modeling Non-functional Requirement

D. Literature Selection

Based on a guide on Walia and Caver there are criteria inclusion and exclusion as an election paper will be discussed [17]:

Inclusion Criteria:

1. The contents of the paper in accordance with to be discussed by reading the abstract research.
2. The publication paper discussed at least 2010 years.
3. Only in the form of a journal or conferences.
4. Paper use English as the introductory language.
5. Include in topic criteria (information system, software engineering, computer science, and informatics, system).
6. Search literature in the international journal of sites basis data.

Exclusion Criteria:

1. Topic not associated with discussion extended use case on the non-functional requirements, and do not cover questions research.
2. Paper does not use English as the introductory language.
3. Besides journals and conferences with the year 2010 rising below years.

E. Extracting Data and Synthesis

The aim of the extraction data is to obtain information accurate and consistent. The data including in the extraction is identification, the name of the writer, the publication, source, reference, methodology data collection, data analysis and concept [19]. The paper also takes additional of reference of main paper who became the idea of making this review literature.

IV. RESULTS AND DISCUSSION

A. Results

Based on searches related to extended use case from various international journals provider sites, the data are obtained and presented in Table 1.

TABLE 1
THE RESULTS OF COLLECTING JOURNALS/PAPERS

IEEE	Science Direct	ACM	Total
1.405	2.590	1.525	5.520

The results of the collecting of journals or papers outlined in table 1 have been through the filtering step by year, the title of the journal entries and the type of content included in the scope of information technology. So, from the total of the results of collecting journals or paper can be classified by the year of publication as shown in Figure 1.

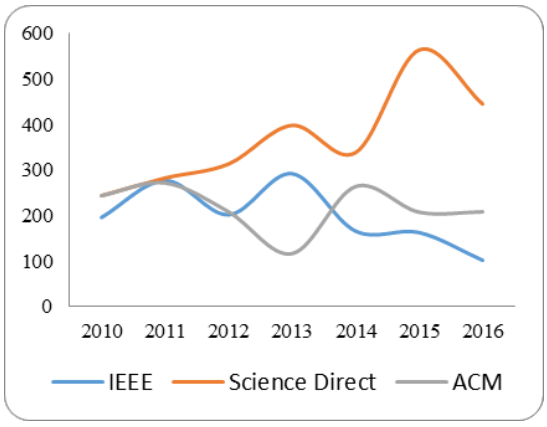

Figure 1. Number of research (from 2010 to 2016)

From the graph above, it can be seen that the most obtained keywords are in Science Direct. However, the most widely discussed in literature come from the IEEE. The following is the result of the selection of papers used in this literature review based on the suitability of the research that is able to answer both predefined questions or the research questions. The filter is done looking at the keywords used and by reading the abstract from the paper that has been obtained. Thus, there is a total of 30 papers referenced in this literature review, as presented in Table 2.

TABLE 2
THE RESULTS OF INCLUSION AND EXCLUSION

Database	Total Retrieved	Final Selection
IEEE	1.405	12
Science Direct	2.590	5
ACM	1.525	9
References from main paper	4	4

B. Discussion

1) RQ1: What is the problem raised by previous research regarding the extended use case in non-functional requirements modeling?

Non-functional requirements are important to define in elicitation step for developing software. When developer defines the requirements of the system, that must include in whole part of requirements. That is why non-functional requirement needs more attention to define in earlier step because if it does not, will be barriers to successful implementation. There is some research that shows problem with the complexity in defining the non-functional requirement. Therefore, in this section will resume the problem related to non-functional requirement modeling. The problem raised to find out the urgency of the importance of modeling non-functional requirements. The Table 3 shows the problem raised by some of related research:

A simple difference between non-functional and functional requirements is the difference in how the system should do something contrasting to what the system will do [23]. Non-functional requirements are rarely defined generally by using high levels, and it needs to be considered to define those requirements in detail while developing the software. Failure to meet non-functional requirements in the software development process will have an impact on the quality of the software, and if improvements are made after the system is implemented it will cost more expensive [24]. There have been many studies that model non-functional requirements because of many types of non-functional requirements that are so complex and difficult to understand [4].

TABLE 3
THE PROBLEM RELATED EXTENDED USE CASE

Problem	Sources
Many types of non-functional requirements (complexity)	[4], [23], [24]
The differences in functionality requirements with non-functional requirements are still unclear	[22], [12]
Abandonment of non-functionality requirement modeling	[10], [22], [11]

The complexity in defining non-functional requirements is due to unclear of the differences with non-functional requirements. The discrepancies are dependent on the details of the requirements document agreed upon by the system owner and system developer. However, the stakeholder's difficulties in expressing non-functional requirements also make developers ignore the modeling of nonfunctional requirements [22]. Nuseibeh said that non-functional requirements are also referred as quality requirements that are generally more difficult to express and measure, so when analyzing it becomes difficult [20]. In non-functional requirements modeling, extended use cases can be a solution of complexity in modeling. This is because extends relationship provides the ability to capture many requirements that can be modeled in the case. Extensions are a real use case but they change the stages in an existing use case. Specifically, extensions are used for specific changes in stages that occur for example such as security, performance, and so on [21].

2) RQ2: Why is an extended use case important to use in non-functional requirements modeling?

A use case is familiar tools for defining requirements when developer develop the system based on what the user needs. The use case has function extended use case that can define such as non-functional requirements. Modeling non-functional

requirements using extended use cases is so important, given the importance of extended use cases in non-functional requirements modeling as shown in Table 4.

TABLE 4
THE IMPORTANCE OF EXTENDED USE CASE IN NON-FUNCTIONAL REQUIREMENT

Importance	Sources
The importance of modeling non-functional requirements from a developer's point of view	[7], [10], [25]
The importance of identifying and defining non-functional requirements in software requirements specification	[8], [11], [12], [26]
The importance of modeling non-functional requirements in UML design	[6], [14], [15], [27], [28]
The importance of using an extended use case in non-functional requirements modeling	[4], [16], [29]
The importance of measuring complexity in non-functional requirements modeling	[4],[14]

Regarding non-functional requirements, non-functional requirements are important to define because these requirements play a vital role in software development [27]. But of course, the definition of non-functional requirements is not easy considering there are 114 non-functional requirements that are defined [30] that lead to complexity in defining. However, when referring to ISO 9126 [7], non-functional requirements can be classified into only 6 major non-functional requirements thus helping the process of defining non-functional requirements and reducing the potential for complexity in non-functional requirements modeling. Based on table 4 related to the importance of extended use case is elaborated on the basis of several points of view from the developer's point of view ([7],[10], [25]) and the analyst's point of view ([8], [11], [12],[26]).

Several methods and strategies have been undertaken to model non-functional requirements to meet non-functional requirements modeling standard [25]. Defined requirements need to be defined by modeling those requirements into UML and need to create schemes for non-functional requirements [26], and one of them is by modeling non-functional requirements using extended use cases [4]. Several studies have been conducted by adapting the concept of extended use cases [29] in non-functional requirement modeling that emphasizes the indicators used to support functional requirements. Extended use cases are easier to use for modeling non-functional requirements because the more familiar use case concepts are used on functional requirements by adding an <<extends>> relationship as well as measured aspects according to non-functional requirements [28], which will make it easier to perform annotations on modeling requirements, especially non-functional requirements. So based on the reason why the extended use case is more suitable for modeling non-functional requirements than other methods is because: 1) an extended use case is easier to use in modeling non-functional requirements; 2) use case is more familiar to use in software requirement modeling; 3) extended use case can minimize errors in software requirement and failure in software development; 4) the ease of annotation in non-functional requirement modeling using extended use case.

3) RQ3: How to use extended use case to model non-functional requirements?

Based on the literature review that explained in the previous section, can be solved the problem of the complexity of defining non-functional requirements using extended use case. So, in this section will give the detail step for modeling non-functional requirements with an extended use case. In this section, the explanation about how to modeling non-functional requirements is described in step by step. The method is not only about the extended use case, but also the other methods will be mentioned in this section. Thus, the difference between extended use case and the others can be considered in modeling non-functional requirements.

There is a semi-supervised approach for modeling non-functional requirements. A semi-supervised approach is natural language description that transformed into the textual specification as known as eliciting step in requirement analysis. There are few steps to model non-functional requirements with a semi-supervised approach as figure 2. First, Learning phase including specification of requirements by categorized the requirement with a label. Then, in the learning phase, Pre-processing is a step for describing how unstructured documents describing requirements are transformed into suitable representations to be used

as input for machine learning algorithms through the application of a number. Then, in the next subsection, To reach the goal (requirements) so the Expectation Maximization (EM) strategy is implemented with naïve Bayesian classifiers. Second, Classification step has classified the category from Semi-Supervised Learning (EM) step. Third, the analysts will receive suggestions about a possible classification of the remaining requirements and they can give feedback to refine classification in an iterative process. Finally, non-functional requirements will be explained in natural language that makes the analyst well-understand about what the exactly non-functional requirement [8]. The advantages of using semi-supervised approach are less effort in labeling requirement, feedback from analyst will improve, help the analyst to manages the requirements. But, the disadvantages of using this approach are the categorized requirement can arise the noise in texts stemming from variations in the vocabulary employed by elicitation teams. Elicitation teams are not always in the same domains, thus semi-supervised classification needs a few labeled requirements that belong to project in progress.

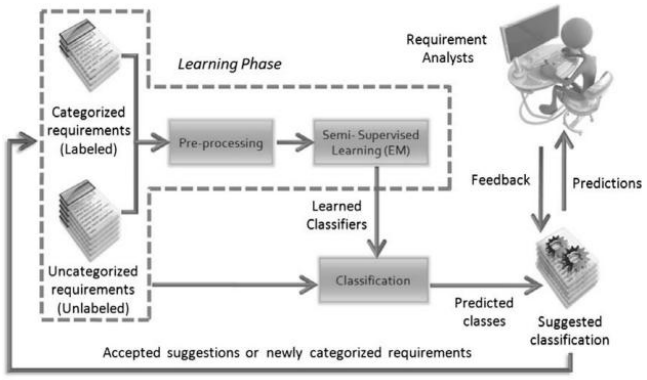

Figure 2. The Semi-Supervised Approach[8]

The other approach for modeling non-functional requirements is classified from SRS Documents and combine with thematic roles. Thematic roles are used to define the thematic relations in the sentences written in natural language. There are some fit criteria for non-functional requirement using IEEE definitions of different NFRs and mapping it with thematic roles. Thus, there is three phase for defining non-functional as described in figure 3.

It starts from giving SRS as an input for the first step of design for document pre-processing. Second, giving the annotation by classifying annotated sentences into various non-functional requirements classes. The third step is classification process by using thematic roles and fit criteria, contains typical text mining operations using ANNIE (a Nearly-New Information Extraction System) components. ANNIE as tools for tokenization process. Then, the resulting sentences are fed to ANNIE POS Tagger for tagging parts of speech, next Multilingual Noun Phrase Extractor (MuNPEx) is used as a noun chunker and along with NE transducer. Then, the other tools such as, ANNIE VP Chunker (for verb groups chunking), GATE Morphological Analyzer, Number Tagger (for tagging numbers), Measurement Tagger (for tagging measurable units), along with ReqGazetteer (contains various gazetteers which help in annotating modality) are used to extract various thematic roles within the requirement documents automatically. Finally, classification takes place using these annotated thematic roles and fit-criteria for each non-functional requirements class [11]. The advantages of using this approach are the identification of non-functional requirements more detail as classified in subclasses. This approach also minimizes the stress and labor of designer and analyst in identifying non-functional requirements. But, the disadvantage of this approach is needed more extra time for classified the non-functional requirements in a large amount of non-functional requirements categories.

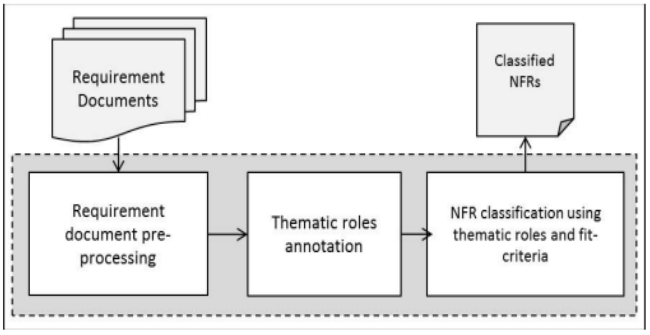

Figure 3. Classification non-functional requirement using SRS document and thematic roles [11]

Non-functional requirements can be also modeled by using a pattern-based approach to design non-functional requirements and integrate the design result into existing functional UML models [13] which is described in figure 4. There are five main steps of pattern-based approach for modeling non-functional requirements which consist of 1) non-functional requirement analysis (identifying and refine non-functional requirements); 2) tactic and patterns-based non-functional requirements design; 3) design model integration; 4) integrating non-functional requirements

related models into existing UML design models; 5) analyze next non-functional requirement tactic.

The advantages of using this method are 1) could be obtained more comprehensive design models which consider both functional requirements and non-functional requirements; 2) by using the patterns of previous design to denote the general design of tactics will be an effective way to reuse the tactic design knowledge. But the disadvantages are: 1) at an early stage before there is a pattern, it is difficult to modeling non-functional requirements using this method; 2) if the pattern is different then it will take a lot of time to modeling it.

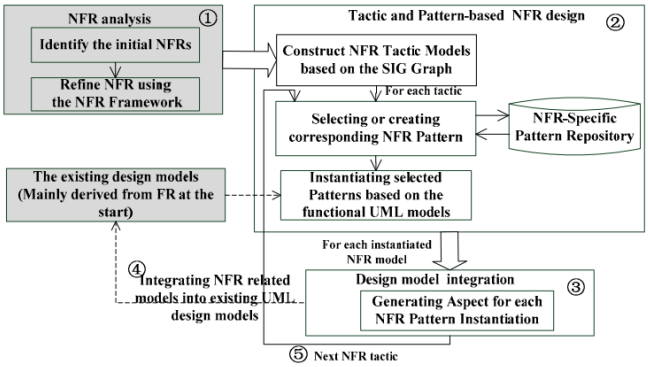

Figure 4. Pattern-based approach to design non-functional requirements [13]

Another method that can be used for modeling non-functional requirements is agile requirements approach [22]. By using this method, not only modeling functional and non-functional requirements of the system from software developer perspective but also modeling business process from the business owner perspective. There are 7 steps in this model as described in figure 5 which consists of 1) enumerate business goals from the need of software; 2) design diagram business process; 3) requirements definition meeting; 4) pattern search; 5) define business process; 6) define functional and non-functional requirements; and 7) pattern addition. The advantages of this approach are: 1) it has allowed better self-expression on the part of the micro-business owner by combining the use of goals, business process model, and non-functional requirements model. 2) requirements documents have been produced quickly without serious compromises by using pattern repository. But the main disadvantage of this method is less specific on modeling non-functional requirements and it will be causing misleading when modeling non-functional requirements using this method.

Non-functional requirements can be also modeled using requirement description schema (RDS) [26]. It is an XML-based versatile specification approach for the structural representation of functional and non-functional requirements. The Requirements can be simple or complex in the sense that they represent the functional and non-functional behavior of a system often requiring an interaction, dependency and priority selection. There are five stages of modeling non-functional requirements using RDS ie: 1) requirement specification with the interchangeable format; 2) mapping requirement engineering phase to design phase; 3) requirement artifacts selection; 4) transformation of requirements to RDS format with GUI; 5) linkage with XML-based security standards. The advantages of using this method are efficient to managing requirement metadata and comprehensive artifacts of requirements like status, priority, version, stability, elicitation source etc. But the disadvantages using this method are: 1) it must understand XML first before modeling the requirements; 2) it is too difficult and takes a lot of time to model more complex requirements.

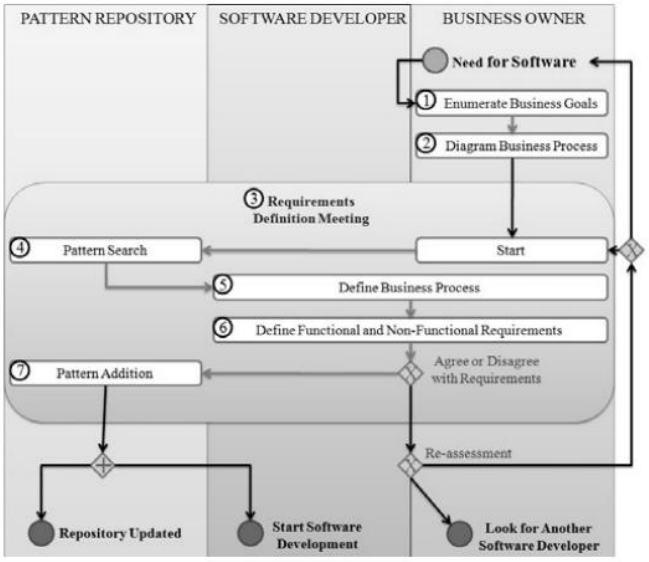

Figure 5. The Agile Requirements Elicitation Approach [22]

Zheng at al proposed an outline to model non-functional requirements using extended use case as described in figure 6 [2]. First, identify the requirements and define actors and use case for initial use case model. The requirements of the system include functional requirement and non-functional requirements. The actors are defined by the summary of their roles in the use case. And the use case itself is commonly defined by using the primary and alternative

scenarios from each actor. Second, from the use case model, is refined by identifying extensions (the non-functional requirements) and inclusions (the functional requirements) of the use case. The use case model is refined by identifying extensions and inclusions and add those refinements on use case model. Third, describe the non-functional requirements at the key association points in the use case model. Non-functional requirements defined as global properties of the system which limit the functional requirement. Fourth, describe architectural policies at platform-independent level through architectural policy scenarios which allow specifying a valid value for each dimension. The last step of the proposed outline is identified aspects in the requirement analysis phase for it makes it possible to begin tackling the problem of tangling scattering of the requirement as early as in requirement analysis phrase.

The security aspect is one of type non-functional requirements. Security requirement can be modeled by an extended use case in 8 step [8]. First, Specify the actors who will be working with the system as a key user. Second, identify dependencies between actors who have different job desk with others, so the developer can find the particular actor. Third, Identify typical objects of the application core. In this step, specified what module or functional that system will have. From that categories, the developer knows how the system works flow. Fourth, Analyze tasks of the actors within the system. In this step, link what the key user's job desk with the application core process, then specify the task for our key user. Fifth, Elaborate use case from the tasks as well as described it in use case diagram. Sixth, map typical objects of the application core to the actors and define access policies. Then, Extend the application core with security aspect such as <<integrity>>, <<secrecy>>, etc. In the end, draw a security extension of the use case into application core which links with actors.

Another modeling approach which developed by [4], proposed 4 steps to modeling non-functional requirements using extended use case. First, identify use cases for the kernel which deals with functional requirements of the system. Kernel captures system functionality from the perspective of actors. The kernel must be kept simple and small by including only use cases with has a direct link to the actor. Second, mapping the non-functional requirement to use case model in the kernel by adding another layer outside the kernel. Third, Identify applicable condition and non-functional requirements case which is black box view of potential non-functional requirements to be defined. An applicable condition is associated with non-functional requirements which include constraint or condition required to implement the non-functional requirement in a system. Last, evaluating the complexity of use case, which defined as the relation of a use case to relationships.

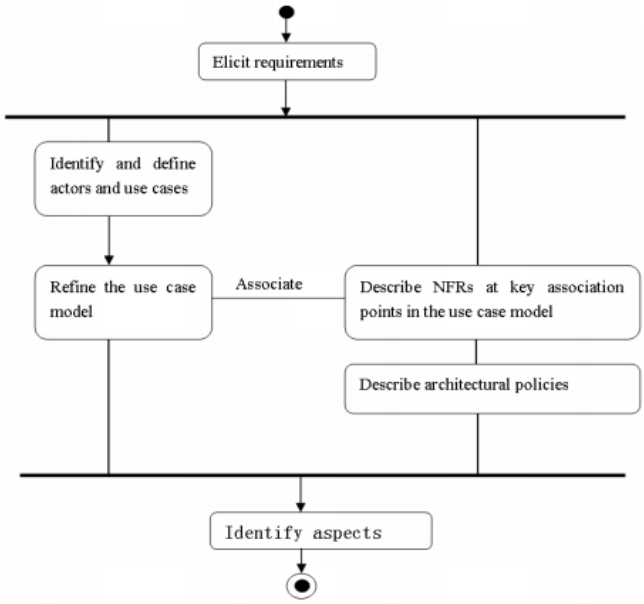

Figure 6. Modeling Non-functional Requirements using Extended Use Case [2]

From the previous research paper, conducted a summary of those papers to modeling non-functional requirements with an extended use case. There are eight step for modeling non-functional requirements with an extended use case which presented in figure 7.

First, identify the requirement of the system that includes functional and non-functional in general. Then, specify the actors that help the developer in eliciting the requirements as known as product champion. Next step, identify dependencies between actors such as actor categories that describe in specific job desk (i.e: admin, teller, etc). Then, identify use cases which deal with functional requirements of the system. Then, define the primary and alternative scenario for each actor such as the primary activity that user do in routined with the system. Next, identify extensions as called as non-functional requirement and inclusion as called as the functional requirement. The difference of this requirement is from the notation of non-functional requirement "<<example>>" beside functional system does not need that notation for modeling the requirements. Then, describe non-functional requirements at the key association points. In the end,

evaluate the complexity of the use case that drew. The complexity of use case comes from the relationships between use case like usage or extension or include or inherit which will add the complexity of use case [4].

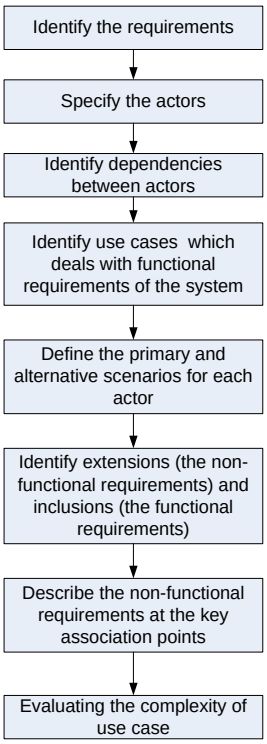

Figure 7. The proposed approach to modeling non-functional requirements using extended use case

4) Non-functional Requirements and Extended Use Case

The importance of modeling non-functional requirements in the elicitation of software requirements and the model which capable of modeling those requirements is extended use case. The advantages of an extended use case in modeling non-functional requirements can minimize errors in software compliance and failure in software development. Before modeling, non-functional requirements must know what non-functional requirements that should be modeled on use case since there are many types of non-functional requirements. The non-functional requirements that can be modeled in the extended use as shown in Table 5.

Based on the search results of international journals database (IEEE, Science Direct, and ACM), the studies related to non-functional requirements modeling using extended use case from year to year shows that the number of research has increased and decreased, especially in 2016 there was a decrease in the number of studies. This decrease in the number of studies provides evidence that more studies have prioritized functional requirements modeling than studies on non-functional requirements modeling. This is supported by studies that have been reviewed that say that non-functional requirements are considered secondary requirements and there is no clear separation between non-functional and functional requirements as it is considered to be a single entity. This is based on a lack of studies on the modeling of non-functional requirements so that the complexity in defining non-functional requirements is still felt by software developers and stakeholders.

TABLE 5
NON-FUNCTIONAL REQUIREMENTS IN EXTENDED USE CASE

Non-functional Requirements	Project	Sources
Security/ security critical system	Internet-based business application	[3],[10]
Availability, performance, robustness, security	Cafeteria ordering system	[4]
Visibility, availability, performance, analyzability, manageability	Dicode (Data-intensive collaboration and decision making) on EU Framework 7 project	[5]
Completeness, correctness, redundancy	1. E-Post Office System 2. Back to My Village 3. Online Project Management 4. Online Job Portal System 5. Real Estate Computer Game 6. Online Shopping Mall 7. Online Stock Market 8. Study System for Educational Institutes	[15]
Security and accessibility	Role-Based Access Control (RBAC) project on the purchasing process	[23]
Adaptivity	News provider services project zan.com	[29]

One method that can be used to model non-functional requirements is the extended use case. A use case represents an interaction between a user (actor) and the system which single unit of use case has a description which describes the functionality of the software system[4]. In the beginning stages of software development, most of the analyst must design the functionality of the system with an use case diagram and it is needed by the programmer to build a system base on that diagram. So it can be assumed that in every software development, developers are accustomed to using an use case diagram to design functional requirements of a system. Besides that, a use case diagram has relationship feature which connects between actor and system or even between use case. In UML defines two types of relationships between use case are include and extend relationships. Two use case is related by the <<extends>> relationship if one use case (know as the base use case) implicitly incorporates the behavior of another use case (know as extension use case which may be contained non-functional requirements) at a specified location [16].
There are some approaches for modeling extended use case, such as a semi-supervised [8], classification non-functional requirement using SRS document and thematic roles [11], a pattern-based [13], agile requirements approach [22] and requirement description schema (RDS) [26]. But, from all of the approach mentioned still have disadvantages, such as need more knowledge or skill to use that approach, less specific on modeling non-functional requirements, and need more extra time for classified the nonfunctional requirements in a large amount of non-functional requirements categories. So based on those explanations it is very easy to use the extended use case in modeling non-functional requirements.

The complexity in defining non-functional requirement is not the necessary problem when using extended use case. From the previous research paper, conducted a summary of those papers to modeling non-functional requirements with an extended use case. There are eight steps for modeling non-functional requirements with an extended use case which presented in figure 2. First, identify the requirement of the system that includes functional and non-functional in general. Then, specify the actors that help the developer in eliciting the requirements as known as product champion. Next step, identify dependencies between actors such as actor categories that describe in specific job desk (i.e: admin, teller, etc). Then, identify use cases which deal with functional requirements of the system. Then, define the primary and alternative scenario for each actor such as the primary activity that user do in routined with the system. Next, identify extensions as called as non-functional requirement and inclusion as called as a functional requirement. The difference of this requirement is from the notation of non-functional requirement "<<example>>" beside functional system does not need that notation for modeling the requirements. Then, describe non-functional requirements at the key association points. In the end, evaluate the complexity of the use case that developer draw. Evaluation of complexity means that developer deal with the stakeholder about elicitation result. Thus, the developer verifies the requirement that stakeholder really needs and already represent what the system needs from both requirements (functional and non-functional).

Related to the problems of non-functional requirements in software development it is necessary to define non-functional requirements in the early stages of software development by specifying non-functional requirements. There are too many kinds of non-functional requirements categories like mentioned before. Besides, non-functional requirements are one of the keys criteria to distinguish between many types of system in the software development. The non-functional requirements that can be modeled with extended use cases are security ([3],[23]), availability ([4], [5]), performance ([4], [5]), robustness [4], visibility [5], analyzability[5], manageability [5], completeness [15], correctness [15], redundancy [15], accessibility [23] and adaptivity ([23] , [29]). So, from many categories of requirements, it has been demonstrated that extended use cases are capable to model non-functional requirements as well as providing ease in understanding the non-functional requirements on the software development phase.

V. CONCLUSION AND FUTURE WORK

One of the causes that make developers and stakeholders ignore the definition of non-functional requirements is because non-functional requirements are considered complex and difficult to translate into an easily understood model. Non-functional requirements are important to be defined in the early stages of software development as well as defining

functional requirements. If non-functional requirements do not define at the early stages of software development then it affects the quality of the software and will take a lot of repair costs after the implementation of the system. Thus, the extended use case is one model that is able to define non-functional requirements that have been proven by previous research. The use of an extended use case makes it easy and understandable because software developers have understood the use case concept and used the <<extend>> relationship to connect it to functional requirements.

There are eight steps for modeling non-functional requirements with an extended use case. First, identify the requirement of the system that includes functional and non-functional requirements. Then, specify the actors. Next, identify dependencies between actors. Then, identify use cases which deal with functional requirements of the system. Then, define the primary and alternative scenario for each actor such as the primary activity that user do in routined with the system. Next, identify extensions as called as non-functional requirements and inclusion as called as the functional requirements. Then, non-functional requirements are described by the key association points. In the end, evaluate the complexity of the use case which has been drawn.

There are two points that we can conclude that the ease of using extended use case in modeling non-functional requirement because the annotation is familiar to use in software development. Besides the complexity of the other approach is difficult to modeling the non-functional requirement ([8], [11], [13], [22], [26]). The complexity in general from that approach are stakeholder and developer need special skill to use that approach and need more extra time to categorized the non-functional requirement. Furthermore, an extended use case can model various categories of non-functional requirements such as security, availability, performance, robustness, visibility, analyzability, manageability, correctness, redundancy, accessibility and adaptivity.

In the future research, the literature review paper on the extended use case in modeling non-functional requirements is expected to trigger the interest of other researchers to develop papers or journals related to non-functional requirements modeling using the extended use case. Furthermore, other studies of complexity that may occur due to the use of extended use cases in other categories as well as attempt to model on other types of projects. Thus, it is expected that in modeling requirements, not just functional requirements that need to be modeled, but also non-functional requirements need to be modeled by using an extended use case.

VI. REFERENCES

[1] M. Rahman, S. Ripon, et al., "Elicitation and modeling non-functional requirements-a pos case study," *arXiv preprint arXiv:1403.1936*, 2014.

[2] X. Zheng, X. Liu, and S. Liu, "Use case and non-functional scenario template based approach to identify aspects," *in Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, vol. 2, pp. 89–93, IEEE, 2010.

[3] G. Popp, J. Jurjens, G. Wimmel, and R. Breu, "Security-critical system development with extended use cases," in *Software Engineering Conference, 2003. Tenth Asia-Pacific*, pp. 478–487, IEEE, 2003.

[4] H. Kaur and A. Sharma, "A measure for modelling non-functional requirements using extended use case," in Computing for Sustainable Global Development (INDIACom), *2016 3rd International Conference on*, pp. 1101–1105, IEEE, 2016.

[5] F. Yang-Turner and L. Lau, "Extending use case diagrams to support requirements discovery," in *Requirements Engineering for Systems, Services and Systems-of Systems (RESS), 2011 Workshop on*, pp. 32–35, IEEE, 2011.

[6] U. I. Hernández, F. J. Á. Rodríguez, and M. V. Martin, "Use processess modeling requirements based on elements of bpmn and uml use case diagrams," in *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on*, vol. 2, pp. V2–36, IEEE, 2010.

[7] J. Zou, L. Xu, M. Yang, X. Zhang, and D. Yang, "Towards comprehending the non-functional requirements through developers eyes: An exploration of stack´ overflow using topic analysis," *Information and Software Technology*, vol. 84, pp. 19–32, 2017.

[8] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information and Software Technology*, vol. 52, no. 4, pp. 436–445, 2010.

[9] X. L. Zhang, C.-H. Chi, C. Ding, and R. K. Wong, "Non-functional requirement analysis and recommendation for software services," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pp. 555–562, IEEE, 2013.

[10] A. Olmsted, "Secure software development through non-functional requirements modeling," in *Information*

Society (i-Society), 2016 International Conference on, pp. 22–27, IEEE, 2016.

[11] P. Singh, D. Singh, and A. Sharma, "Classification of non-functional requirements from srs documents using thematic roles," in *Nanoelectronic and Information Systems (iNIS), 2016 IEEE International Symposium on*, pp. 206–207, IEEE, 2016.

[12] A. Mahmoud, "An information theoretic approach for extracting and tracing non-functional requirements," in *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pp. 36–45, IEEE, 2015.

[13] Y. Liu, Z. Ma, R. Qiu, H. Chen, and W. Shao, "An approach to integrating non-functional requirements into uml design models based on nfr-specific patterns," in Quality Software (QSIC), 2012 *12th International Conference on*, pp. 132–135, IEEE, 2012.

[14] R. K. Chopra, V. Gupta, and D. S. Chauhan, "Experimentation on accuracy of non-functional requirement prioritization approaches for different complexity projects," *Perspectives in Science*, vol. 8, pp. 79–82, 2016.

[15] S. Tiwari and A. Gupta, "Does increasing formalism in the use case template help?" in *Proceedings of the 7th India Software Engineering Conference*, p. 6, ACM, 2014.

[16] M. Misbhauddin and M. Alshayeb, "Extending the uml use case metamodel with behavioral information to facilitate model analysis and interchange," *Software & Systems Modeling*, vol. 14, no. 2, pp. 813–838, 2015.

[17] O. Rebollo, D. Mellado, and E. Fernández-Medina, "A systematic review of information security governance frameworks in the cloud computing environment.," *J. UCS*, vol. 18, no. 6, pp. 798–815, 2012.

[18] Y. Hakami, S. Tam, A. H. Busalm, and A. C. Husin, "A review of factors affecting the sharing of know ledge in social media.," *Science International*, vol. 26, no. 2, 2014.

[19] A. S. Pillai et al., "A study on the software requirements elicitation issues-its causes and effects," in *Information and Communication Technologies (WICT), 2013 Third World Congress on*, pp. 245–252, IEEE, 2013.

[20] K. Alghathbar, "Representing access control policies in use case.," *Int. Arab J. Inf. Technol.*, vol. 9, no. 3, pp. 268–275, 2012.

[21] M. R. Dube and S. K. Dixit, "Extended behavioral modeling using structured use cases and profiles," in *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, pp. 737–740, ACM, 2010.

[22] R. Macasaet, L. Chung, J. L. Garrido, M. Noguera, and M. L. Rodríguez, "An agile requirements elicitation approach based on nfrs and business process models for micro-businesses," in *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, pp. 50–56, ACM, 2011.

[23] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are" non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," in *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pp. 832–842, IEEE, 2016.

[24] A. Buarque, J. Castro, and F. Alencar, "The role of nfrs when transforming i* requirements models into oo-method models," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 1305–1306, ACM, 2013.

[25] Y. Bin, J. Zhi, and C. Xiaohong, "An approach for selecting implementation strategies of non-functional requirements," *in Proceedings of the Fourth Asia-Pacific Symposium on Internetware*, p. 20, ACM, 2012.

[26] T. Shah and S. Patel, "A novel approach for specifying functional and non-functional requirements using rds (requirement description schema)," *Procedia Computer Science*, vol. 79, pp. 852–860, 2016.

[27] J. Eckhardt, D. M. Fernández, and A. Vogelsang, "How to specify non-functional requirements to support seamless modeling? a study design and preliminary results," in *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on*, pp. 1–4, IEEE, 2015.

[28] M. Asadi, S. Soltani, D. Gasevic, M. Hatala, and E. Bagheri, "Toward automated feature model configuration with optimizing non-functional requirements," *Information and Software Technology*, vol. 56, no. 9, pp. 1144–1165, 2014.

[29] M. Luckey, B. Nagel, C. Gerth, and G. Engels, "Adapt cases: extending use cases for adaptive systems," in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 30–39, ACM, 2011.

[30] D. Mairiza, D. Zowghi, and N. Nurmuliani, "An investigation into the notion of non-functional requirements," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, pp. 311–317, ACM, 2010.