Applying Information Retrieval Technique for Security Requirements Verification based on Security Patterns

Chatchapong Changadwech, Nakornthip Prompoon

***Abstract*—Identification of system security requirements is one of the important quality requirements in the system development. In generally, system security requirements include specific functions for preventing possible threats may occur to the system. Specifying security requirements to cover organization security policy and best practice is a difficult task since it requires a high level of security knowledge, understanding and application. Security patterns generally define security problems and solutions that can be applied in various security contexts but they were not presented in a form that can be directly applied in a system analysis and design phase. This research presents a set of UML use case descriptions templates constructed from security patterns and related researches for the verification of the use case description created by a developer. The store and retrieval of the proposed use case template was applied information retrieval technique based on Vector Space Model with elements weight. The verification was performed into 2 aspects of completeness criteria: steps of workflow and part of speech sentence matching. The result of this research provides recommendations on the unfulfilled part of the security context so that security requirements can be improved for a better coverage of security patterns. Examples of how the proposed verification method works are also given.**

***Index Terms*—Security Patterns, Security Requirements, Use Case Description**

I. Introduction

SOFTWARE analysis and design are important steps in the process of software development since the main product results, software requirements and design specification, are significantly used for software validation and verification. One of the major software quality concerns is security requirements [1] which normally embedded as a part of software functions and included in software requirements and design specification in order to ensure that a system will be assessed by ones who were authorized and a system will be developed to meet user security requirements.

Manuscript received December 29, 2015; revised January 09, 2016. This work was supported in part by the Graduate School, Department of Computer Engineering, Chulalongkorn University

C. Changadwech is with the Software Engineering Lab, Center of Excellence in Software Engineering, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand; e-mail: Chatchapong.C@student.chula.ac.th).

N. Prompoon is with the Software Engineering Lab, Center of Excellence in Software Engineering, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand; e-mail: Nakornthip.S@chula.ac.th).

For example, Microsoft [2] recognizes the importance of security requirements through the inclusion of security assurance process as part of its system development. Security Development Lifecycle (SDL) was implemented with embedded security and privacy policies in Microsoft software and corporate culture.

During the software analysis and design phase, security requirements should be defined to cover the necessary security criteria and design goals which are normally defined from organization security policy and standard. However, enforcing security policy and needs in an analysis and design phase can be difficult for developers as they must study and understand best practices of security concepts and techniques as well as being able to apply such knowledge to actual environment.

The security patterns proposed by Schumacher et al [3] are ones that help developer resolve this problem since they provided the proposed security solutions in various groups of patterns for reuse purpose. The proposed patterns covered security body of knowledge in confidentiality, integrity, authorization and accountability aspects. They may be applied in security requirements specification, design and implementation. The content of each pattern was covered existing security problems, solutions, and examples of its application in various contexts for developer to select the appropriate one for a particular security problem.

UML (Unified Modeling Language) Use Case diagram and Use Case description are widely used among software developers to identify software function or service and interaction between actor or external agent and software. Use Case description, especially in normal flow and alternative flow section is used to explain a sequence of steps on all feasible scenarios using natural language. They are usually used between users and developer team to define and clarify all possible ways of interaction in both normal and exceptional events of system functions. In the same manner, a developer may also identify system security feature using Use Case and Use Case description. Writing comprehensive use case description with accurate sequence of actions for all possible scenarios can be difficult for developer since it requires detail of business context understanding in which software has to operate. In addition, it requires intensive knowledge for a particular application domain such as right and role identification for information asset access. For example, patient medical record privacy of a hospital information system is a critical security issue.
The system must have a service to define information access

right, such as only a doctor can access his/her patient record.

This research proposes a method to define a description of system security by applying security patterns context and relevant researches using Use Case and Use Case description in order to help developer verify and define a system security feature created by developer.

The content of the paper is as follows: Section 2 provides related work. Section 3 provides an overview of security patterns. Section 4 provides an overview of use case and description. Section 5 presents our proposed method. Section 6 presents our approach for security use case description definition. Section 7 presents security requirements verification. Finally, Section 8 presents conclusion and future work.

II. Related Work

A number of researches have proposed guidelines for using security patterns as basic knowledge in defining security requirements. Research [4] analyzed security elements and security pattern relationship to create a security grammar that can define security requirements in natural language and developed an automatic security support tool. Research [5] presented a guideline on how to apply security patterns and security use case to define functional requirements, which could be done through a modification of the security requirements defined by the security grammar proposed by Research [4] into a semi-formal language. UML Diagram was used to describe software functions with the use of Use Case Diagram and Use Case description. Sequence Diagram was used to show a sequence of message among system objects for each Use Case. Sequence Diagram and the resulting objects were explained with the use of Class Diagram and description to support natural language security requirements development, which may become ambiguous when used in the analysis and design of software security functioning. Research [6] proposed a guideline for the creation of an automatic use case description based on relevant problem framework of existing problems. It is necessary to define the needs to solve such problems by using a use case description template to define the needs which were classified by types of problem frameworks involved. Research [7] proposed a method to extract Access Control Policy (ACL) from documented natural language software before using the obtained data to define formal requirements through synthetic and meaning analyses of the natural language data, and through the extraction of data that revealed ACL's sequence of actions. The evaluation results revealed that the method could create an understanding of how the method could be applied to define organizational policies.

From analyzing these related researches, there is a gap to fulfill in term of system security definition for helping developer specify and design security context for a similar problem needed to be solved. Thus, our research statement is whether there are a set of meta-processes of proven solutions for a security context.

III. Security Patterns

Security patterns [3] present previous security problems and solutions proposed by experts in security, security engineering and software engineering. To derive a solution guideline to these problems, 46 relevant security patterns were divided into 8 groups. These patterns are very helpful for software developers as they can be used to develop suitable software for a specific context as well as to endorse the best practices of a system design and development.

The security pattern elements that reveal characteristics of a security pattern are represented by the pattern template such as Name, Also Known As, Example, Context, Problem, Solution, Structure, Dynamic, Implementation, Example Resolved, Variants, Know Uses, Consequence and See Also. Security patterns can be applied to encourage identification of software security needs. This research has adopted these security patterns to support the formulation of a set of security use case descriptions combining them with security information context from relevant researches [4, 5].

IV. Use Case and Description

UML Use case [8] was used to define software requirements that reflected interaction between a user and a system. Identification of actor and system interaction sequences to realize individual function was indicated in the use case descriptions. Examples of the use case description template created in this research were presented to describe content of a security requirements template for a security meta-process with its elements titles and description shown in Table I.

TABLE I
Use Case description template

Use case name	Name of use case.
Primary actor	The name of the actor who participate in use case.
Brief description	A description showing how this use case adds value to the organization.
Precondition	The conditions that must be satisfied before this use case can be invoked.
Trigger	The event that causes the use case to begin.
Normal flow of events	The individual steps within the business process are described.
Subflow	In some cases, the normal flow of events should be decomposed into a set of subflows to keep the normal flow of events as possible.
Alternative or exceptional flow of events	Alternative or exceptional flows are once that do happen but are not considered to be normal case.
Postcondition	The condition that will be established as a result of invoking this use case successfully.

V. Proposed Method

The work accomplished by this research is divided into two major parts, as shown in Figure 1. The first part is to develop a security process template using UML use case description. The proposed template can be used to verify the use case descriptions described by software developers. It comprises three work steps: (1) Study security patterns, (2) Generate a security use case description template, and (3) Evaluate security use case description with the help of security experts. The second part is the application of security use case descriptions to verify security requirements created by developer. This part comprises three work steps: (1) Use Case description template storage and retrieval, (2) Security components Extraction from both use case

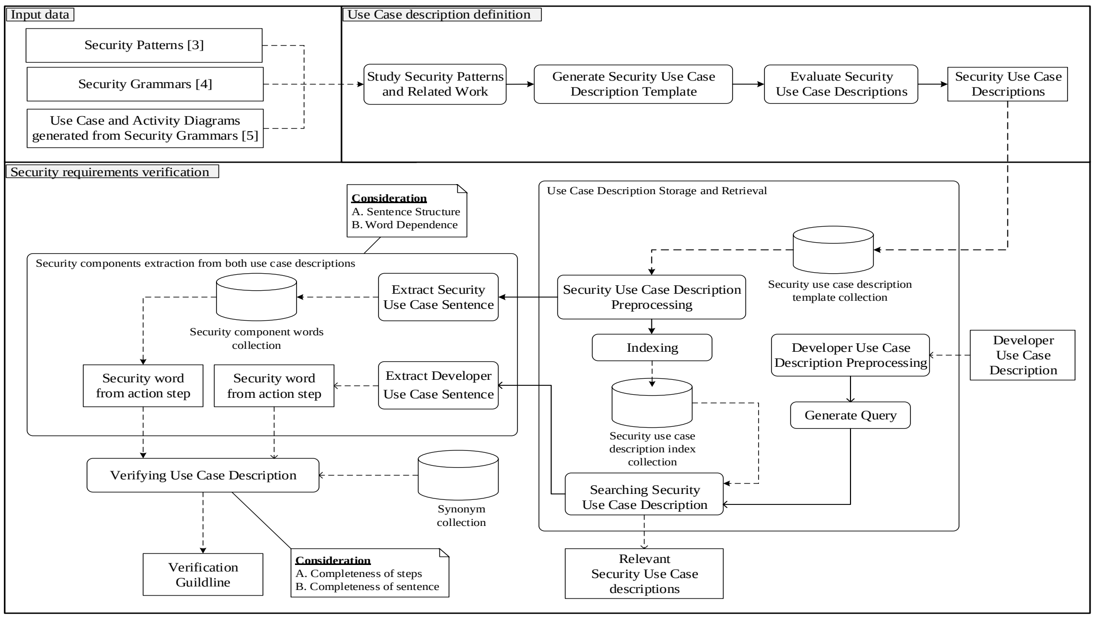
Input data
Security Patterns [3]
Security Grammars [4]
Use Case and Activity Diagrams generated from Security Grammars [5]
Use Case description definition
Study Security Patterns and Related Work
Generate Security Use Case Description Template
Evaluate Security Use Case Descriptions
Security Use Case Descriptions
Security requirements verification
Consideration
A. Sentence Structure
B. Word Dependence
Use Case Description Storage and Retrieval
Security use case description template collection
Security components extraction from both use case descriptions
Extract Security Use Case Sentence
Security Use Case Description Preprocessing
Security component words collection
Indexing
Developer Use Case Description Preprocessing
Developer Use Case Description
Security word from action step
Security word from action step
Extract Developer Use Case Sentence
Generate Query
Security use case description index collection
Verifying Use Case Description
Synonym collection
Searching Security Use Case Description
Verification Guildline
Consideration
A. Completeness of steps
B. Completeness of sentence
Relevant Security Use Case descriptions

Fig. 1. The Proposed Method Overview for Security Use Case Defining from Security Patterns and Use Case Verification Method

descriptions, and (3) Verification of security use case description which were defined by developers against the proposed security meta-process defined as a use case description and finally present result of verification findings.

VI. Security Use Case Description Definition

In creating security use case description as a meta-process template, it is essential to study each component of a security pattern to understand its context and application in details and the associated research papers [4, 5]. These research papers illustrated the application of security patterns which were used to fulfill each pattern objective and context. The resulting of security was evaluated by security domain experts. Its feedbacks were analyzed and used to improve to derive a quality template that meets the specified security context criteria. The following steps were taken to create a security use case description template:

A. Study Security Patterns

This research covered 20 patterns from 4 groups of security patterns composed of Enterprise Security and Risk Management, Identification and Authentication, Access Control Models and Firewall Architecture since they are normally concerned as a basis of security system policies and system requirements and they have a high level of inter-relationship for security application domain.

The Reference Monitor Pattern, which belongs to a group of security patterns for access control models, was chosen as a case study for assets request from users to illustrate the meta-process described in the proposed use case description. It is necessary for the system to verify the right of access role before accepting or rejecting such a request. These data were analyzed and sequenced appropriately with a major concern for a general application for any domain. Scenario-based covering in use case steps was the main goal must be accomplished. Thus, some major elements of security pattern show runtime behavior pattern such as Structure, Dynamic, Implementation and Example Resolve together with the result from research [4, 5] findings were also analyzed for creating security scenario. The result of this step was the security context understanding of each pattern especially the problem and the solution.

B. Generate a Security Use Case Description Template

Building a security use case description template began with defining security components in terms of natural language syntax and semantic for a sentence of use case description elements; primary actor, precondition, brief description, trigger, normal flow, subflow, alternative flow and postcondition in order to satisfy the pattern objective.

For the purpose of applying the proposed a security use case description template that corresponds to a particular security pattern, natural language sentences in major elements of it was defined by applying the UML stereotype and natural language structure. The defined stereotype in each sentence can be later on replaced with the actual environment application. Each sentence was created based on [4, 5]. Data from analyzing the proposed security grammar [4] generated from component analysis of natural language sentences in security patterns, were resulted as terms and part of speech were used to define specific context of the work steps. The requirements identification for specifying which user (<Subject Name>) who has a role (<Role Name>) and want to send a request (<Right>) to access a system asset (<Protection Object>) will be in a form as: <Subject Name> has <Role Name> sends request to <Right> <Protection Object>.

A sequence of step in normal flow, subflow and alternative flow was generated from a corresponding UML Sequence diagram from [5]. This template was then used to verify the use case descriptions defined by the developer whether they conformed to the structured components.

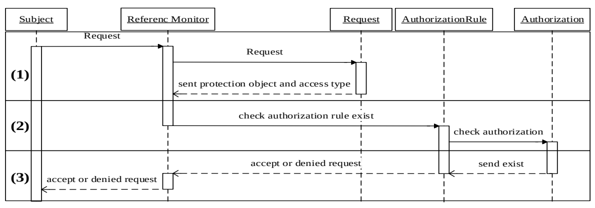

Fig. 2. A simple sequence diagram generated from reference monitor security pattern [5]

Figure 2 shows the sequence of steps of reference monitor pattern that was consistent with the security pattern in this case study research example. The sequence diagram shows the primary workflow which consist of: (1) A request for access to the Projection Object, (2) Examination of user's right to access the Projection Object, and (3) Reporting acceptance/termination of access request to users

The final step was to define the use case security description after having analyzed security components from the findings of Research [4, 5]. The researcher had created a use case description with a consideration to the following components: use case name, primary actor, brief description, precondition, trigger, normal flow of events, subflow, alternative or exceptional flow of events and postcondition. This use case description also contained the security components that were consistent with the security pattern and the sequence of steps in a specific security pattern context. The security sentences in this use case description are shown in Table II.

TABLE II
A SECURITY USE CASE DESCRIPTION TEMPLATE GENERATED FROM REFERENCE MONITOR SECURITY PATTERN

Use Case Name	Check authorization role
Primary Actor	<Subject Name>
Brief Description	This use case describes subject's role is authorized
Precondition	<Role Name> has been defined (from SEC_UC: 'Define role right' was defined from authorization pattern in same group)
Trigger	1. <Subject Name> has <Role Name> sends request to <Right> <Protection Object>
Normal flow of events	1. <Subject Name> has <Role Name> sends request to <Right> <Protection Object> 2. <Reference Monitor System> asks for <Role Name> information to access <Protection Object> 3. <Subject Name> enters information 4. <Reference Monitor System> verifies <Role Name> information using <Authorization Rule> 5. <Reference Monitor System> allows access to request <Protection Object> 6. <Reference Monitor System> sends allow message to <Subject Name>
Subflow	-
Alternative or exceptional flow of events	4a: Invalid <Subject Name> has <Role Name> <Right> 4a.1 <Reference Monitor System> denies access to <Right><Protection Object> 4a.2 <Reference Monitor System> sends deny message to <Subject Name> 4.a.3 End
Postcondition	<Subject Name> can <Right> <Protection Object>

Some elements from security pattern such as <Subject Name>, <Right>, <Role Name>, <Reference Monitor System> and <Protection Object> are used to determine the meaning of domain specific words. For example, <Subject Name> is Bob, <Role Name> is professor, <Right> is update, <Reference Monitor> is system and <Protection Object> is subject data.

C. *Evaluate Security Use Case Description*

This step involved an evaluation of the use case description template which was created prior to its application. Overall and specific evaluations of each use case description were carried out by security experts in terms of its accuracy, completeness, benefit, and satisfaction of application to security operation. Evaluation feedbacks were used to improve the security use case description template.

VII. SECURITY REQUIREMENTS VERIFICATION

Using use case description to verify natural language security requirements of both the research's template and the developer's template, the template used for verification had to be defined first. Since there are a number of use case templates, Vector Space Model was used for information retrieval and identification of the desired use case template based on its relevant to the use case that needed verification. Once the verification use case was identified, a representative of the document extracted from the key words in both documents was then created as tokens whose functions were compared in two contexts of: 1) Word meaning and tokens interdependence, and 2) The order of tokens appearing in a sentence. The researchers proposed an approach to store and retrieve the documented security use case descriptions and their verification, which consisted of the following steps:

A. *Use Case Description Template Storage and Retrieval*

All proposed use case description templates were stored and pre-processed as index inverted files which collected terms and their frequency of occurrences in each element of the template. An inverted file will be used as use case description templates representation and will be used to retrieve the relevant security document templates which is similar to a specify query [9]. In this research context, a query represents use case description developed by a developer and needs to be verified with the proposed security template. Query representation was earned in a similar fashion by extracting term in each template element. Vector Space Model with elements weight was used to retrieve the relevant proposed security templates. The major steps for storing and retrieving security use case descriptions are as follows [10, 11]:

Storage of documented use case descriptions

The security use case description template created by this research was stored and indexed for subsequent retrieval purpose as an index inverted file.

Retrieval of documented use case descriptions

Retrieval of documents relating to user's queries on relevant use case descriptions. The retrieval results displayed as a group of documented security use case descriptions related to the queries based on Vector Space Model principles as shown in equation (1) and (2) where each use case description components were considered.

$$Similarity(UC_i, Querty_j) = \frac{\sum_{m=1}^{e}[Similarity(E_{mi} \cdot E_{mj}) \cdot WE_m]}{TotalWeight} \qquad (1)$$

Where *Similarity*(*UC*$_i$, *Query*$_j$) is similarity scores between each element of user's use case query and each element of each use case collected in the collection, *WE*$_m$ is weighted value of element *m* defined by the user, *TotalWeight* is summary value of weighted values (*WE*$_m$) of all components.

Where *Similarity*(E_{mi}, E_{mj}) is similarity score between each element of use case *i* and query use case *j* computed by equation (2).

$$Similarity(E_{mi}, E_{mj}) = \frac{2[\sum_{m=1}^{e}(Term_{mik} \cdot Term_{ik}) \cdot Term_{mjk}]}{\sum_{k=1}^{t} Term_{mik} + \sum_{k=1}^{t} Term_{mjk}} \qquad (2)$$

Where E_{mi} is element *m* of use case *i*, E_{mj} is element *m* of use case *j*, W_{ik} is IDF weighted value of term *k* in use case *i*, $Term_{mik}$ is 1 when term *k* appears in element *m* of use case *i* and 0 when term *k* does not appear in element *m* of use case *i*, $Term_{mjk}$ is 1 when term *k* appears in element *m* of use case *j* and 0 when term *k* does not appear in element *m* of use case *i*.

B. Security Components Extraction from Both Use Case Descriptions

Three element parts: normal flow, subflow and alternative flow, of use case description from developer and the relevant proposed use case description template were preprocessed and compared using the following steps: Since use case descriptions are in natural language sentences, contents with specific relation to a security pattern needs to be extracted from specific word groups for subsequent processing. The extraction process started with a preparation of pre-processing data through word segmentation, word normalization, and pattern normalization. The resulting data were then extracted from natural language sentences with security grammar through the process of 1) Grammatical analysis of natural-language sentences, 2) Using Stanford Parser [12, 13] to analyze word interdependence. The results from this step were extracted data from the use case descriptions in the desired structure. Examples of data extraction process through the use case description created in this research are:

Using english grammar to analyze natural-language sentences in use case descriptions

Vocabularies related to the interested security functions were analyzed and the results were used to verify the completeness of work sequences during the verification phase of the use case description characteristics. Examples of English grammatical analyses of natural sentences are shown in Table III.

TABLE III
USE CASE FLOW SENTENCE (FROM TABLE II) STRUCTURES ANALYSED BY NATURAL LANGUAGE PROCESSING TOOL [12]

Normal flow of events
1. Bob/NNP has/VBZ professor/NN sends/VBZ request/NN to/TO update/VB subject/NN data/NNS
2. System/NNP asks/VBZ for/IN professor/NN information/NN to/TO access/VB subject/NN data/NNS
3. Bob/NNP enters/VBZ information/NN
4. System/NNP verifies/VBZ professor/NN information/NN using/VBG authorization/NN rule/NN
5. System/NNP allows/VBZ access/NN to/TO request/NN subject/NN data/NNS
6. System/NNP sends/VBZ allow/VB message/NN to/TO bob/NN
Alternative or exceptional flow of events
4a.1 System/NNP denies/VBZ access/NN to/TO update/VB subject/NN data/NNS
4a.2 System/NNP sends/NNP deny/VB message/NN to/TO Bob/NNP

Analysis of word interdependence in use case description sentences

This was an analysis of the interdependence of security-related words in which the results were used for verification of data as shown in Table IV.

TABLE IV
THE WORD DEPENDENCIES IN USE CASE STEP SENTENCE (SENTENCE 1 FROM TABLE II) REPRESENTED BY NATURAL LANGUAGE PROCESSING TOOL [13]

nsubj(has-2, Bob-1)
root(ROOT-0, has-2)
dobj(has-2, professor-3)
acl:relcl(professor-3, sends-4)
dobj(sends-4, request-5)
mark(update-7, to-6)
advcl(sends-4, update-7)
compound(data-9, subject-8)
dobj(update-7, data-9)

C. Verification of the Security Use Case Description

The functioning of developer's use case descriptions were verified by its sequences of steps and security components of each action in relation to one defined in the proposed security template. The verification results were evaluated in the normal flow, subflow and alternative flows of the use case description, focusing on the verbs used in the security use case description sentences created by developer and one by the proposed template. Examples of developer's functional use case descriptions are shown in Table V. IEEE 830 Recommended Practice for Software Requirements Specifications [14] was used to verify the defined characteristics of the use case description. Verification was performed in two aspects:

TABLE V
A USE CASE DESCRIPTION FROM DEVELOPER [15]

Use case name	Login
Primary actor	The actor
Brief description	This use case describes how a user logs into the Course Registration System.
Trigger	This use case starts when the actor wishes to log into the Course Registration System.
Normal flow of events	1. The system requests that the actor enters his/her name and password. 2. The actor enters his/her name and password. 3. The system validates the entered name and password and logs the actor into the system.

Completeness of steps in all three flows

Verification of steps completeness in developer's use case descriptions within the context of security descriptions was performed based on verb part matching. Examples of this operation are shown in Table VI.

TABLE VI
A SEQUENCE ACTION VERB IN USE CASE DESCRIPTION COMPLETENESS VERIFICATION

Security use case sequence of steps from Table III	**Developer's use case sequence of steps from Table V**
1. has/VBZ, sends/VBZ 2. asks/VBZ, access/VB **3. <u>enters/VBZ</u>** **4. <u>verifies/VBZ</u>, using/VBG** 5. allows/VBZ 6. sends/VBZ, allow/VB 4a.1 denies/VBZ, update/VB 4a.2 sends/VBZ, deny/VB	1. requests/VBZ enters/VBZ **2. <u>enters/VBZ</u>** **3. <u>validates/VBZ</u>, logs/VBZ**
<u>Outcome</u> It can be seen that Actions 1, 2, 4a.1, 4a.2, 5, and 6 are still missing from the software developer's use case description in the context of security descriptions in relation to the right of system access or assets access. <u>Since the verb 'verifies' is similar in meaning to the verb 'validates' it is necessary to define word groups of similar meaning in order to improve verification efficiency.</u>	

Table VI indicates the deliberation of characteristics completeness through the verification of processed verbs in natural language, which can increase the verification efficiency of security sequence of steps in the use case description. From the verification example, it shown that there were only two verbs matched and the alternative flow part was missed.

Completeness of a selected sentence

A matching sentence of the specify step which is the result from previous step was selected to check the completeness of the sentence components. According to the example, based on synonym, sentence in the 3rd step of the developer used case and 4th step of the proposed use case template was selected for component completeness checking.

TABLE VII
A NORMAL STEP SENTENCE IN USE CASE DESCRIPTION CORRECTNESS VERIFICATION

The 4th normal step sentence from Table III	**The 3rd Normal step sentence of developer's use case from Table V**
System/NNP verifies/VBZ **professor/NN** information/NN using/VBG **authorization/NN rule/NN**	The/DT system/NN validates/VBZ the/DT entered/JJ name/NN and/CC password/NN and/CC logs/VBZ the/DT actor/NN into/IN the/DT system/NNP
Outcome It can be seen that the actions in Step (4) of the security use case description and Step (3) of the developer's use case description as verified in Table 6 are similar due to the use of similar verbs, 'verifies' and 'validates'. When the security component was considered, the said actions were not a confirmation of the right to access as the verb 'verifies' is written under the control of the words 'Professor' and 'Authorization rule'.	

Table VII indicates the missing part of speech in the 3rd step of a developer use case which is user's role or <Role Name>. It is a security component of access control defined in the security pattern.

Nevertheless, these verification outcomes led to recommendations on the above-mentioned on two non-compliance characteristics. For example, it was recommended that developer should define user role and authorization rules correspond with the user role. Thus the developer may use the recommendation for improving the security use case description.

VIII. CONCLUSION AND FUTURE WORK

The research proposed a set of use case description templates as a meta-process used for documented use case descriptions verification written by software developers. The proposed templates were based on a study and an analysis of existing security patterns and related researched [4, 5] that have been created to satisfy the defined security purposes. The proposed templates were stored and retrieved for verification purpose using information storage and retrieval technique based on Vector Space Model with elements weight. The most relevant template retrieved was used to verify the completeness of the use case description developed by a developer in two aspects; step completeness and sentence completeness. The step completeness was verified based on the number of verbs matching criteria with the use of synonym. The sentence verification was based on the completeness of the sentence components. The outcome of the research is the missing part of use case description compared with the proposed use case description templates which developer may use it as a feedback for use case description improvement.

It is hoped that this proposed use case description templates will be used by developers as a tool for an analysis and design of system security operation and as a guideline for the identification of the operation that conforms organization security goal, target environment and goals of security patterns.

In the future, a supporting tool will be developed to enhance an automatic verification between use case descriptions that will save verification time.

REFERENCES

[1] C.H. Jane, "Software Requirements", School of Computer Science, Telecommunications, and Information System, DePaul University.
[2] Microsoft. (2015, August). Security Development Lifecycle. [Online]. Available: https://www.microsoft.com/en-us/sdl/
[3] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad, "Security Patterns: Integrating Security and System Engineering", John Wiley & Sons, 2005.
[4] K. Supaporn, N. Prompoon and T. Rojkangsadan, "Enterprise Assets Security Requirements Construction from ESRMG Grammar based on Security Patterns", *APSEC 2007. 14th Asia-Pacific*, 2007, pp. 112-119.
[5] N. Tikhumpornpittaya and N. Prompoon, "Transformation of Security Requirements into Use Case Diagram and Sequence Diagram using Security Grammar and Security Metadata", Master's Thesis, Department of Computer Engineering, Faculty of Computer Engineering, Chulalongkorn University, 2012.
[6] A. Fatolahi, S.S. Some and T.C. Lethbridge, "Automated Generation of Use Case Descriptions from Problem Frames", *Eighth ACIS International Conference on Software Engineering Research, Management and Applications*, 2010, pp. 223-230.
[7] X. Xiao, A. Paradkar, S. Thummalapenta and T. Xie, "Automated Extraction of Security Policies from Natural-Language Software Documents", *FSE '12 Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, Article 12, 2012.
[8] David Tegarden, Alan Dennis and Barbara Haley Whixord, "Systems Analysis and Design with UML", April 2012.
[9] A. Udomchaiporn, N. Prompoon and P. Kanongchaiyos, "Software Requirements Retrieval Using Use Case Terms and Structure Similarity Computation", *XIII Asia Pacific Software Engineering Conference (APSEC'06)*, 2006, pp. 113-120.
[10] M.J. McGill and G. Salton, Introduction to modern information retrieval, McGraw-Hill, 1983.
[11] B. Ribeiro-Neto and R. Baeza-Yates, Modern information retrieval, Addison-Wesley, 1999.
[12] The Stanford NLP Group. (2015, October). Stanford Parser. [Online]. Available: http://nlp.stanford.edu:8080/parser/
[13] The Stanford NLP Group. Dependencies. (2015, October). [Online]. Available: http://nlp.stanford.edu/software/stanford-dependencies.html
[14] IEEE-SA Standard Board, IEEE Recommended Practice for Software Requirements Specification, 1998.
[15] Mastering OOAD with UML. (2003, February). Course Registration System Use-Case Model. [Online] Available: http://faculty.ksu.edu.sa/abdksu/IS%20496%202008/04course_reg_uc_model_rpt.pdf