A Domain Ontology Based Approach to Identify Effect Types of Security Requirements upon Functional Requirements

Bilal Al-Ahmad, Kenneth Magel, and Sameer Abufardeh

***Abstract*—Inter-requirements traceability refers to finding the *relationships* between requirements. Several approaches have been identified cooperative, conflicting, and irrelevant relationships between requirements. However, the current solutions have a lack of capturing the syntactic and semantic aspects of requirements, and less attention has been paid to relating security requirements with functional requirements.**

To overcome these limitations, we propose to use a domain ontology based approach, in which a domain ontology can be used as a domain knowledge to discover relationships between requirements. Our proposed solution is a hybrid approach which uses: 1) a syntactic parsing technique to decompose the requirements statements into Subject, Verb, and Complement constructs, 2) a domain ontology to create a knowledge repository about security and functional requirements concepts, and 3) a rule based system to build several detection rules that identify security requirements effects upon functional requirements. We evaluate our approach in a case study of requirements for an online medical database system that shows how the effect types can be determined.

***Index Terms*—Detection rules, domain ontology, effect types, security requirements.**

I. Introduction

As confusing requirements often increases the cost of software development projects, requirements analysis is considered as the most critical phase in software development life cycle. Requirements analysis concentrates on developing high quality requirements specification that are satisfactory to the customers and feasible to the developers. Inter-requirements traceability [1] focuses on linking requirements with other requirements. An example of Inter-requirements traceability is to find the cooperative, conflicting, and irrelevant relationships. Inter-requirements traceability has significantly influenced a number of different activities during software development such as consistency checking and impact change.

Security requirements are considered as restrictions or constraints on system functionalities such as user identification, secure access, secure storage, and secure communication. More specifically, security and functional requirements have been defined by different studies. For example, Haley [2] described the security requirements as a part of non-functional requirements that constrained the functional requirements of a system. Also, Kotonya and Sommerville [3] define security requirements as restrictions or constraints on system services. In addition, Rushby [4] defined security requirements as common concerns of the system that must not occur.

Functional requirements [5] are defined as the requirements that describe the system behavior by expressing it as the inputs to the system, the outputs from the system, and the relationships between inputs and outputs. Because the security requirements are considered as constraints on the functional requirements, we called relationship types as effect types.

Domain ontology is a very popular semantic processing technique that classifies concepts and relations among concepts within a particular domain. Domain ontology is an essential element that can be used to obtain great success in requirements analysis phase since it lets us have a semantic source for requirements descriptions which supports better understanding of relationships between software requirements.

Using domain ontology in requirements engineering helps to capture the requirements information, as well as reuse and share concepts and relations that are represented by the ontology. In this paper, we propose a domain ontology based traceability approach that helps to identify cooperative, conflicting, and irrelevant effects between security and functional requirements.

II. Background and Related Work

Identification of cooperative, conflicting, and irrelevant relationships between software requirements is very important because it affects several significant software activities such as consistency checking and impact of requirements change. Reviewing the literature, we found that several studies [6]-[9] applied different approaches to identify cooperative, conflicting, and irrelevant relationships among software requirements.

Egyed and Grünbacher [6] developed an approach for determining conflicts and cooperation dependencies among software requirements based on using both quality attributes of requirements and the automated traceability technique. Also, Liu [7] identified the conflicts and cooperation dependencies among uncertain software requirements based on using a fuzzy logic technique.

Temponi, Yen, and Tiao [8] employed Quality Functional Deployment (QFD) methodology to translate customer

Manuscript received November 14, 2014; revised January 16, 2015.

Bilal Al-Ahmad was with Jordan. Now he is with Software Engineering at North Dakota State University, USA (e-mail: bilal.alahmad@ndsu.edu).

Kenneth Magel and Sameer Abufardeh are with North Dakota State University, USA.

satisfaction (i.e., customer requirements) into organization functions (i.e., technical requirements), and applied fuzzy logic based requirements analysis to represent QFD since it can handle the fuzzy expressions in requirements.

Lee and Xue [9] used a goal based approach to explore cooperative, conflicting, and irrelevant relationships between user requirements. They represented the user requirements by building the use case models with the associated goals. But the current approaches have several limitations: only considering fuzzy requirements, neglecting the syntactic and semantic features of software requirements, and slight attention has been given to show the effects of security requirements upon functional requirements.

III. The Domain Ontology Based Approach

To decrease the ambiguity and inconsistency of the informal natural language of requirements, the proposed approach captures both syntactic and semantic features of requirements statements. The syntactic aspect of requirements focuses on grammatical analysis of requirement statement constructs, while the semantic aspect of requirements focuses on understanding the meaning of requirements. Fig. 1 shows the proposed methodology.

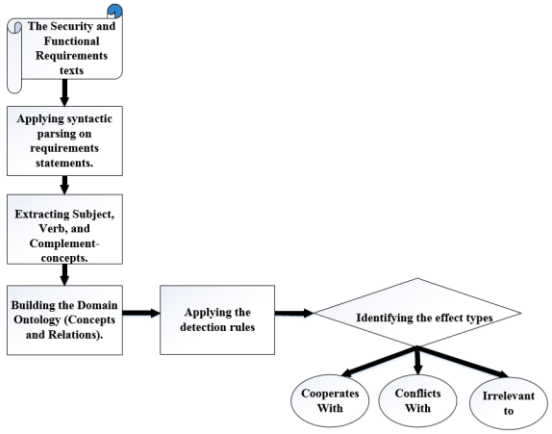

Fig. 1. An overview for our domain ontology detection approach.

In our approach, we integrate syntactic analysis (i.e., syntactic parsing) and semantic analysis (i.e., domain ontology). Our proposed approach includes the following four main steps:

1) Applying the syntactic parsing on both security and functional requirements statements by using a parsing tool [10] to split each requirements statements into Subject, Verb, and Complement constructs. Also, we consider each requirement construct as a single concept. Each requirement statement has three concepts: Subject, Verb, and Complement.
2) Building the domain ontology to represent the domain concepts and relations for a particular domain. Security and functional requirement concepts have three types [11]:

Subject-concept (which represents requirement entity), Verb-concept (which represents requirement action), and Complement concept (which represents the extra description for the requirement entity, or the Object that represents requirement target). The relations initially have seven types: Generalization, Aggregation, Association, Synonyms, Antonyms, Identical, and No-relation}.

Fig. 2 shows the proposed domain ontology, in which the first three relations (i.e., Association, Generalization, and Aggregation) are extracted from the class diagram, Synonyms and Antonyms relations are extracted from WordNet lexical database [12], and the last two relations (i.e., Identical and No-relation) used to represent identical matching and no-matching relations between concepts.

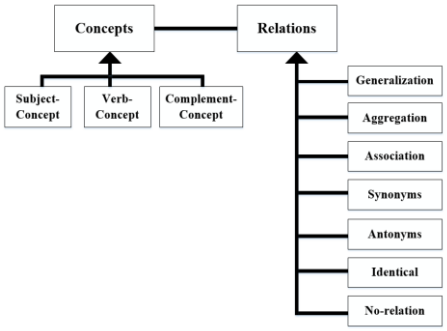

Fig. 2. Domain ontology for our proposed approach.

1) Generating detection rules. Each detection rule includes three conditions (i.e., three relations between security and functional requirement concepts) and single conclusion (i.e., single effect type). Effect can be one of following types: Cooperates with, Conflicts with, and Irrelevant to. Every ontology has an engine and the rule based system considered as the engine for the domain ontology.
2) Identifying effect-types of Security Requirement (SR) upon Functional Requirement (FR) based on the proposed detection rules.

Our approach consists of two important parts: (1) Security Functional Tracing Model (SFTM), and (2) Security Functional Requirements Diagram (SFRD). SFTM is the proposed tracing model for both security and functional requirements while SFTM is a requirements diagram that illustrates the effects of security requirements upon functional requirements.

A. Detection Rules

Requirements often conflict with each other, particularly those from various perspectives. Furthermore, many conflicts among requirements are difficult to detect.

As a result, we propose to use novel detection rules that can help to identify conflicts between these security and functional requirements.

These detection rules are constructed using IF-THEN rules [13]. Every detection rule has three relations and one effect type. The first relation combines two Subject-concepts, the second relation has two Verb-concepts, and the third relation has two Complement-concepts. The structure for the proposed detection rule is as follows:

IF *Relation* (Subject-concept (SR), Subject-concept (FR))
AND *Relation* (Verb-concept (SR), Verb-concept (FR))
AND *Relation* (Complement-concept (SR), Complement-concept (FR))
THEN *Effect-type* ∈ {Cooperates with, Conflicts with, Irrelevant to}

There are several detection rules that determine each effect type, it. In the requirements examples listed below, Verb-concept is formatted as italic. The following detection rules were developed to determine the cooperative, conflicting, and irrelevant effects between security requirement and functional requirements:

1) Cooperative effect

In this type of effect, the functional requirement is positively affected by the security requirement. Thus, both security and functional requirement can be implemented at the same time. For example:

SR: The user can *only read* the student's financial history.

FR: The employee can *retrieve* the student's payment history.

The following detection rule identifies the cooperative effect:

> IF *Generalization* (user, employee)
> AND *Synonyms* (only read, retrieve)
> AND *Generalization* (student's financial history, student's payment history)
> THEN *Effect-type*∈ {Cooperates with}

2) Conflicting effect

Here, the functional requirement is negatively affected by the security requirement. Thus, both security and functional requirement cannot be implemented at the same time. For example:

SR: The registrar *cannot change* the student's academic records.

FR: The user *can modify* the student's grades.

The following detection rule identifies the conflicting effect:

> IF *Generalization* (registrar, user)
> AND *Antonyms* (cannot change, can modify)
> AND *Aggregation* (student academic records, student grades)
> THEN *Effect-type*∈ {Conflicts with}

3) Irrelevant effect

Lastly, in this effect the functional requirement is neither positively nor negatively affected by the security requirement. The implementation of the security requirement does not affect the implementation of the functional requirement.

B. Security Functional Tracing Model

We propose a tracing model called Security Functional Tracing Model (SFTM) that shows a complete tracing of all security and functional requirements. This model will help a requirements engineer to detect the effects between all security and functional requirements.

This model consists of four essential subsets: security requirements, functional requirements, tracing and effect types. Each set is a subset of the superset SFTM. SFTM for system x is described as follow:

1) Security Requirements set, denoted as: $SR_{(x)}$ = {SRi, SRi+1, …, SRn}, where x is the name of software system.
2) Functional Requirements set, denoted as: FR(x) = {FRj, FRj+1, …, FRm}.
3) Tracing Pairs set, denoted as T(x) ={ (SRi, FRj), (SRi, FRj+1), …, (SRi, FRm), (SRi+1, FRj),(SRi+1,FRj+1), …, (SRi+1, FRm), …, (SRn, FRj),(SRn, FRj+1), …, (SRn, FRm)}.
4) Effect types set, denoted as E(x) = {Cooperates with, Conflicts with, and Irrelevant to} Then, SFTM will be defined as: $SFTM_{(x)} = \{SR_{(x)}, FR_{(x)}, T_{(x)}, E_{(x)}\}$.

C. Motivation Example

If there is a system A with three security requirements named (SR1, …, SR3) and six functional requirements named (FR1, …, FR6). Then, the proposed tracing approach is to map the first security requirement (SR1) with all six functional requirements to get the following tracing pairs {(SR1, FR1),…, (SR1, FR6)}. Next, the second security requirement (SR2), until the third security requirement SR3 proceeds through the tracing of the six functional requirements in order to obtain these tracing pairs: {(SR3, FR1),…, (SR3, FR6)}. For each pair, the effect type will be identified based on the predefined detection rules. Fig. 3, Fig. 4, and Fig. 5 illustrate the tracing process and the associated effect types for SR1, SR2, and SR3 respectively.

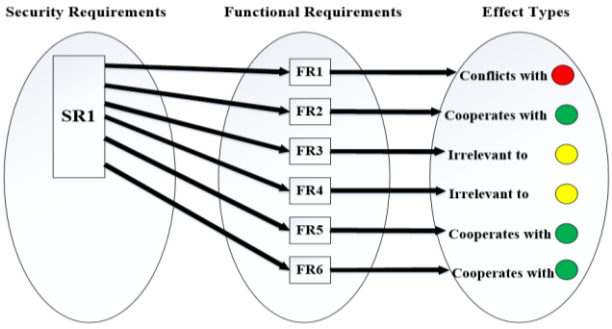

Fig. 3. Security functional tracing of SR1 for system A.

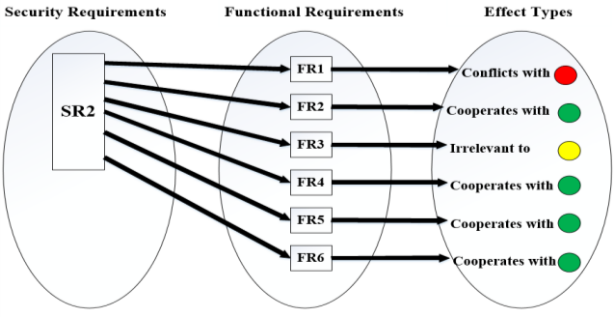

Fig. 4. Security functional tracing of SR2 for system A.

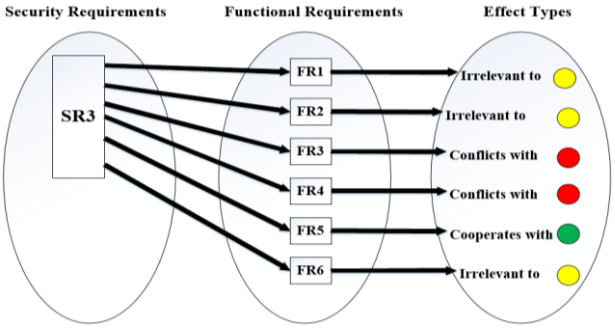

Fig. 5. Security functional tracing of SR3 for system A.

Based on the proposed model, system A will be described as follows:

1) $SR_{(system\ A)}$ ={ SR1, SR2, SR3}

2) $FR_{(system\ A)}$ = { FR1, …, FR6}
3) $T_{(system\ A)}$ = {(SR1, FR1), …, (SR1, FR6),…,(SR3, FR1), …, (SR3, FR6)}.
4) $E_{(system\ A)}$ = {Conflicts with, …, Cooperates with , …, Irrelevant to, …, Irrelevant to }

Consequently, SFRD will be graphed as in Fig. 6, SFRD shows the effect types of security requirements upon functional requirements. This diagram has the security requirement at the top and a node for each functional requirement on the left. These nodes are color-coded to indicate whether or not they are affected by the security requirement. Different types of effects (i.e., relationships) are indicated by different colors. The Cooperative effect is shown in green, the conflicting effect is shown in red, and the irrelevant effect is shown in yellow.

Fig. 6. SFRD for system A.

IV. The Importance of Investigating the Effect Types

A. Prioritization of Security Requirements

One of the significant uses for the proposed approach is to prioritize security requirements based on their weight. Weight is calculated by counting the number of the functional requirements that cooperate with the security requirement. For example, system A has three security requirements (SR1, SR2, and SR3). As in Fig. 7, SR1 has three cooperated functional requirements (FR2, FR5, and FR6). Also, SR2 has four cooperated functional requirements (FR2, FR4, FR5, and FR6) and SR3 has only one cooperated functional requirement (FR5).

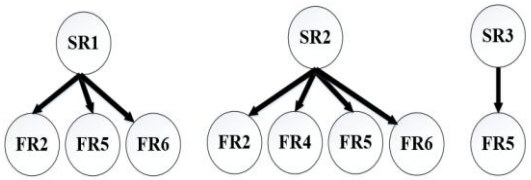

Fig. 7. The cooperated functional requirements nodes with SR1, SR2 and SR3 for system A.

Then, the weight for each security requirement is calculated like:

- Weight (SR1) = 3
- Weight (SR2) = 4
- Weight (SR3) = 1

Therefore, SR2 has the highest priority, and SR3 has the lowest priority. The security requirements prioritization is considered to be significant for the requirements selection task.

B. Finding Association between Security Requirements

In the proposed approach, we used the Jaccard similarity technique [14] to measure the association between the security requirements nodes using the following formula:

$$\text{Asscoiation (SRi, SRj)} = \frac{SRi \cap SRj}{SRi \cup SRj} \quad (1)$$

where:

- SRi, SRj: represents security requirements.
- SRi ⋂ SRj: represents the number of joint cooperated functional requirements nodes between SRi and SRj.
- SRi ⋃ SRj: represents the number of all cooperated functional requirements nodes in both SRi and SRj.

By applying the Jaccard similarity for the example in Fig. 6, the resulting association values are as follows:

- Association (SR1, SR2) = 3/4= 0.75
- Association (SR1, SR3) = 1/3= 0.33
- Association (SR2, SR3) = 1/4= 0.25

Fig. 8 represents shows a weighted graph for the association among SR1, SR2, and SR3. The association between SR1 and SR2 is 0.75, which reflects strongly connected requirements while the association between SR2 and SR3 is 0.25, which reflects weakly connected requirements.

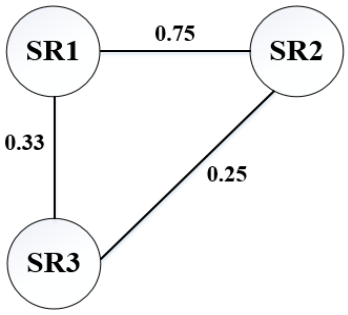

Fig. 8. A weighted graph showing the association among security requirements SR1, SR2, and SR3 of system A.

V. Case Study

In this case study, we demonstrate the effectiveness of our approach in the context of an Online Medical Database system (OMD). This system provides an interesting challenge because it is rich in different semantic relations. Moreover, the privacy and security issues are critical to such systems. To briefly explore the proposed approach, we trace the three Security Requirement (SR1, SR2, and SR3) with 18 Functional Requirements (FR1, …, FR18). The security and functional requirements statements are parsed by applying the syntactic parser. To process requirements semantically, we build the domain ontology for both security and functional requirements.

The domain ontology helps to create knowledge base about the requirements concepts and relations; this leads to an increase of the recognition of the requirements information. The domain ontology is represented in the form of a class diagram, which includes a single class for each concept. Concept can be represented as either a single term or a phrase. To indicate the type of concept, Subject-concept,

Verb-concept, and Complement-concept are stereotyped as <<Subject>>, <<Verb>>, and <<Compl>> respectively.

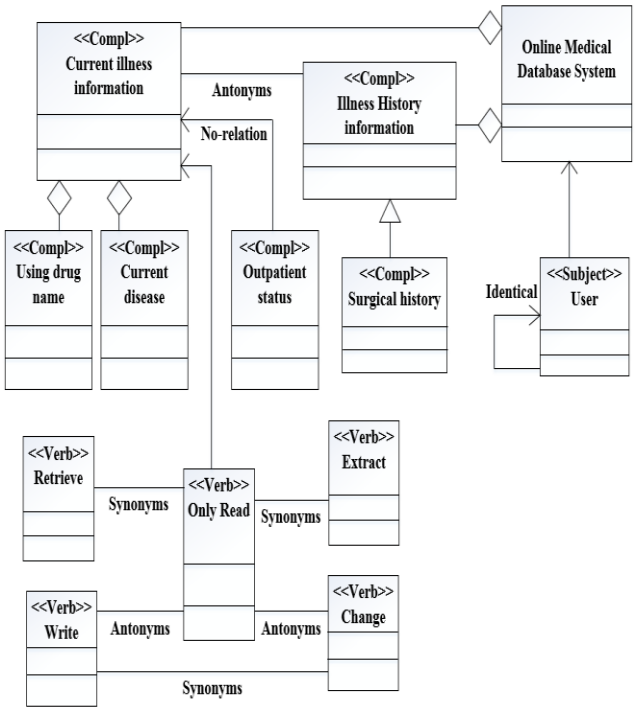

Fig. 9. A part of domain ontology for OMD system.

Fig. 9 shows a portion of the domain ontology for OMD system. Based on the resulting domain concepts and relations that we obtained from domain ontology, we constructed detection rules to identify the effect types between security and functional requirements. We show an example for the effect types and the corresponding detection rules of SR1 as follows:

1) SR1: User can *only read* current illness information.
2) FR4: The user can *extract* the outpatient status.

IF *Identical* (user, user)
AND *Synonyms* (only read, extract)
AND *No-relation* (current illness information, outpatient status)
THEN *Effect-type* ∈ {Irrelevant to}

3) SR1: User can *only read* current illness information.
4) FR5: The user can *retrieve* the using drug name.

IF *Identical* (user, user)
AND *Synonyms* (only read, retrieve)
AND *Aggregation* (current illness information, using drug name)
THEN *Effect-type* ∈ {Cooperates with}

5) SR1: User can *only read* current illness information.
6) FR10: The user can *modify* current diseases.

IF *Identical* (user, user)
AND *Antonyms* (only read, modify)
AND *Aggregation* (current illness information, current diseases)
THEN *Effect-type* ∈ {Conflicts with}

The effect types have been identified based on the detection rules, then SFRD can be illustrated as in Fig. 10

According to the SFRD, we can find clearly the cooperated functional requirements nodes for the security requirements as follows:

1) SR1 has four cooperated functional requirements: FR5, FR6, FR7 and FR8.
2) SR2 has six cooperated functional requirements: FR3, FR5, FR6, FR7, FR8 and FR10.
3) SR3 has ten cooperated functional requirements: FR5, FR6, FR7, FR8, FR9, FR11, FR12, FR14, FR15, and FR17.

Therefore, the weight for each security requirement is calculated as follows:

- Weight (SR1) = 4
- Weight (SR2) = 6
- Weight (SR3) = 10

Based on the weight values, SR3 has the highest priority, and SR1 has the lowest priority. In addition, by applying the Jaccard method, the obtaining association values between the security requirements of OMD system are as follows:

- Association (SR1, SR2) = 4/6 = 0.66
- Association (SR1, SR3) = 4/10 = 0.40
- Association (SR2, SR3) = 4/12 = 0.33

Fig. 11 is a weighted graph that shows the association among SR1, SR2, and SR3. The association between SR1 and SR2 is 0.66, which reflects strongly connected requirements while the association between SR2 and SR3 is 0.33, which reflects weakly connected requirements. These association results will help the requirement engineer to figure out the strength of dependencies among the security requirements.

Functional Requirements (FR)	Security Requirements (SR)		
	SR1	SR2	SR3
FR1	🟡	🟡	🟡
FR2	🟡	🟡	🟡
FR3	🔴	🟢	🔴
FR4	🟡	🟡	🟡
FR5	🟢	🟢	🟢
FR6	🟢	🟢	🟢
FR7	🟢	🟢	🟢
FR8	🟢	🟢	🟢
FR9	🟡	🔴	🟢
FR10	🔴	🟢	🔴
FR11	🟡	🔴	🟢
FR12	🟡	🔴	🟢
FR13	🟡	🔴	🔴
FR14	🟡	🔴	🟢
FR15	🟡	🔴	🟢
FR16	🟡	🔴	🔴
FR17	🟡	🔴	🟢
FR18	🟡	🔴	🔴

Fig. 10. SFRD of OMD system.

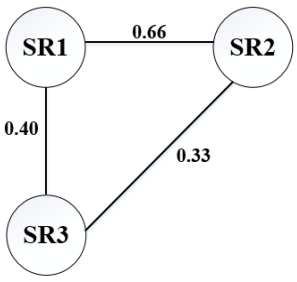

Fig. 11. A weighted graph showing the association among security requirements SR1, SR2, and SR3 of OMD system.

The additional valuable usage that we can get from using SFRD is to find the inconsistency ratio for each security

requirement. The inconsistency ratio is calculated by considering all the conflicting effects relative to the total number of effects. For example, as in Fig. 10, we can find the conflicting effects number for the security requirement as follows:

- Number of conflicting effect for SR1 = 2
- Number of conflicting effect for SR2 = 9
- Number of conflicting effect for SR3 = 5
- Total number of effects for each security requirement = 18

Then, the inconsistency ratio for each security requirement is calculated as follows:

- Inconsistency ratio (SR1) = 2/18 = 11%
- Inconsistency ratio (SR2) = 9/18 = 50%
- Inconsistency ratio (SR3) = 5/18 = 27%

In this case, SR2 has the highest inconsistency ratio while SR1 has the lowest ratio.

VI. Conclusion and Future Work

In this paper, we introduced a new hybrid traceability approach for using syntactic parsing, domain ontology, and a rule based system. This proposed approach helps to identify cooperative, conflicting, and irrelevant effects of security upon the functional requirements. Our approach offers several benefits: 1) it serves as a structured mechanism to simplify the finding of effects, 2) it bridges the gap between functional and non-functional requirements, and 3) it supports the requirements analysis to improve consistency between conflicting requirements. The proposed model will be further extended by developing a tool that will help automatically generate all the possible combinations for concepts and relations in both security and functional requirements, automate the construction process of detection rules, and generate SFRD.

References

[1] F. A. C. Pinheiro, "Requirements traceability," in *Perspectives on Software Requirements*, J. C. Sampaiodo Prado Leite and J. H. Doorn, Eds., The Netherlands: Kluwer Academic Publishers, 2004, ch. 5, pp. 91-113.

[2] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh, "Arguing satisfaction of security requirements," in *Integrating Security and Software Engineering: Advances and Future Visions*, H. Mouratidis and P. Giorgini, Eds., Hershey, PA: Idea Group Publishing, 2007, ch. 2, pp. 16-43.

[3] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, New York: Wiley, 1998.

[4] J. Rushby, "Security requirements specifications: How and what?" presented at Symposium on Requirements Engineering for Information Security, Indianapolis, 2001.

[5] R. Malan and D. Bredemeyer. (June 1999). Functional requirements and use cases. [Online]. Available: http://www.bredemeyer.com/pdf_files/functreq.pdf

[6] A. Egyed and P. Grünbacher, "Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help," *IEEE Software*, vol. 21, no. 6, pp. 50-58, Nov.2004.

[7] X. F. Liu, "Fuzzy requirements," *IEEE Potentials*, vol. 17, no. 2, pp. 24–26, May 1998.

[8] C. Temponi, J. Yen, and W. A. Tiao, "House of quality: A fuzzy logic-based requirements analysis," *European Journal of Operational Research*, vol. 117, no. 2, pp. 340–354, Sep. 1999.

[9] J. Lee and N. L. Xue, "Analyzing user requirements by use cases: A goal-driven approach," *IEEE Software*, vol. 16, no. 4, pp. 92-101, July 1999.

[10] D. Temperley, D. Sleator, and J. Lafferty. (2014). Link Grammar Parser. Carnegie Mellon University. [Online]. Available: http://www.link.cs.cmu.edu/link/submit-sentence-4.html

[11] X. F. Liu, K. Noguchi, and W. Zhou, "Requirement acquisition, analysis, and synthesis in quality function deployment," *International Journal of Concurrent Engineering: Research and Applications*, vol. 9, no. 1, pp. 24–36, March 2001.

[12] C. Fellbaum. (2010). About WordNet. WordNet. Princeton University. [Online]. Available: http://wordnet.princeton.edu

[13] M. Sasikumar, S. Ramani, S. M. Raman, K. Anjaneyulu, and R. Chandrasekar, *A Practical Introduction to Rule Based Expert Systems*, New Delhi, India: Narosa Publishing House, 2007, ch. 7, pp. 129-130.

[14] S. Niwattanakul, J. Singthongchai, E. Naenudorn, "Using of Jaccard coefficient for keywords similarity," in *Proc. the International Mult.i Conference of Engineers and Computer Scientists*, pp. 380-384, vol. 1, Hong Kong, 2013.

Bilal Al-Ahmad received the B.Sc. degree in computer information systems from Jordan University of Science & Technology, Jordan in 2006, the master's degree in computer information systems from Yarmouk University, Jordan in 2009. Currently he is a PhD student in software engineering at North Dakota State University, USA. Before starting the PhD, he has been involved in teaching and research in software engineering at the Hashemite University, School of Information Technology, Jordan.

Kenneth Magel earned his PhD from Brown University, USA in 1977. He joined the Computer Science Department at North Dakota State University since 1983. Currently he is an associate head for Computer Science Department at North Dakota State University, USA. Before joining North Dakota State University, he has taught in several computer science schools in Kansas, Missouri, and Texas, USA. His teaching interests includes courses in problem solving, software engineering, human-computer interaction, object-oriented systems, and programming languages. His research interest's focuses on what makes programming difficult and programs complex, and he has published widely in the areas of Program Complexity Metrics and Software Testing.

Sameer Abufardeh received the B.Sc. degree in computer science from Southern Illinois University, USA in 1994, the master's degree in computer science from St. Cloud State University, USA in 2000, and PhD Degree in software engineering from North Dakota State University, USA in 2009. Currently he is an assistant professor of Practice in Computer Science Department at North Dakota State University. He has published several papers in software localization, software testing, global software development, and natural language processing.