DOMAIN ONTOLOGY BASED DETECTION APPROACH TO IDENTIFY EFFECT TYPES OF SECURITY REQUIREMENTS UPON FUNCTIONAL REQUIREMENTS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Bilal Ibrahim Al-Ahmad

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Program:
Software Engineering

May 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

DOMAIN ONTOLOGY BASED DETECTION APPROACH TO IDENTIFY EFFECT TYPES OF SECURITY REQUIREMENTS UPON FUNCTIONAL REQUIREMENTS

By

Bilal Ibrahim Al-Ahmad

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Kenneth Magel
Chair

Sameer Abufardeh

Simone Ludwig

Gokhan Egilmez

Approved:

5/7/2015
Date

Brian Slator
Department Chair

ABSTRACT

Requirements engineering is a subfield of software engineering that is concerned with analyzing software requirements specifications. An important process of requirement engineering is tracing requirements to investigate relationships between requirements and other software artifacts (i.e., source code, test cases, etc.). Requirements traceability is mostly manual because of difficulties automating the process. A specific mode of tracing is inter-requirements traceability, which focuses on tracing requirements with other requirements. Investigating inter-requirements traceability is very important because it has significant influence on many activities of software engineering such as requirements implementation, consistency checking, and requirements impact change management. Several studies used different approaches to identify three types of relationships: cooperative, conflicting, and irrelevant. However, the current solutions have several shortcomings: (1) only applicable to fuzzy requirements, user requirements, and technical requirements, (2) ignoring the syntactic and semantic aspects of software requirements, and (3) little attention was given to show the influence of security requirements on functional requirements. Furthermore, several traceability tools have a lack of using predefined rules to identify relationships.

To overcome these limitations, our approach uses a rule based system to construct several deterministic detection rules that identify relationship types between security and functional requirements. Our proposed approach has two main parts: (1) Security Functional Tracing Model (SFTM), and (2) Security Functional Requirements Diagram (SFRD). To apply SFTM and generate SFRD, we developed a tool called Detection Rules Constructor (DRC). The experimental results shows that our approach is very effective for automating the tracing of security requirements with functional requirements.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Kenneth Magel, for his excellent supervision, incredible encouragement, extreme support, and insightful comments in every step of my research which enabled me to complete successfully my dissertation.

Special thanks to my supervisory committee, Dr. Sameer Abufardeh, Dr. Simone Ludwig, and Dr. Gokhan Egilmez for their support, guidance, and helpful suggestions. Without their comments and assistance this dissertation would not have been successful.

The sincere thankfulness are due to my mom, Ali’a, and my dad, Ibrahim, for their praying, endless love, and wonderful supporting throughout my whole life. This accomplishment is impossible without them. I would like to thank my wife, Ala’a, for her inspiration and patience that allowed me to complete my graduate study. Special thanks to my beloved daughter, Leen.

Lastly, I wish to thank my dear wonderful brothers, Nidal and Ihap, and all of my favorite friends for being an endless source of encouragement through my graduate study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1. INTRODUCTION

This research is related to requirements engineering, requirements traceability, security and functional requirements, domain ontology, and rule based system. In this chapter, we briefly introduce each of these areas.

1.1. Requirements Engineering

Requirements engineering [1] is a first phase of software life cycle that is concerned with the use of systematic and repeatable techniques that ensure the completeness, consistency, and relevance of the system requirements as in Figure 1. Thus, requirements engineering [2] must identify the software functionalities and the constraints on how the software must be designed and implemented. Requirements engineering includes five main activities; requirements elicitation, requirements analysis, requirements specification, requirements verification, and requirements management. The final output of requirements engineering is the software requirements specification (SRS), which is a document that clearly describes each requirement's functions, performance, design constraints, and quality attributes of the software.

There is an increasing awareness of the importance of requirements engineering, based on empirical investigations and industry experiences, many studies [3, 4, 5] emphasized that the requirements engineering process is an essential contributor to the overall quality of software. As a significant process of software engineering, requirements engineering [6] plays a critical role throughout the whole software engineering lifecycle.

A major problem is that poorly defined requirements often lead to the failure of a software project. As a result, well-defined requirements will increase the possibility of the overall success of a software project.

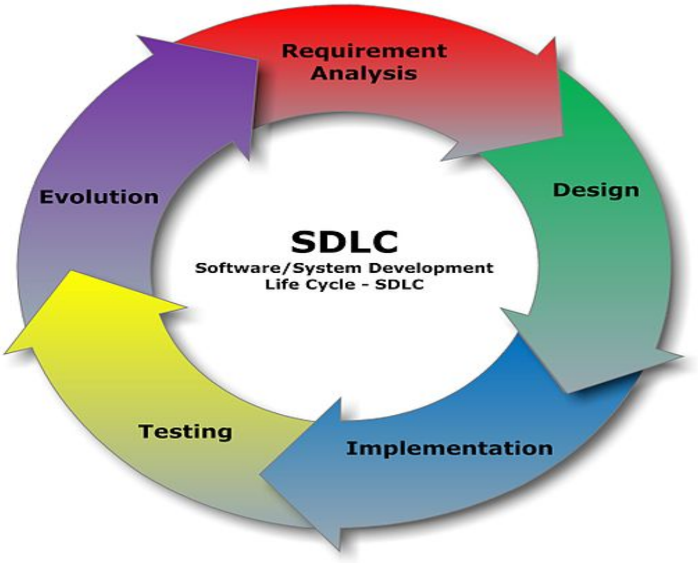


Figure 1. Software development life cycle

1.2. Requirements Traceability

Requirements traceability is an important activity of the requirement engineering process that concentrate on finding relationships between requirements, or between requirements and other software artifacts as in Figure 2. Requirements traceability [7] is defined as the process of linking requirements with either design, implementation components (i.e., forward direction) or with its source's (i.e., backward direction). Requirements traceability has a significant impact on different activities of software engineering process such as requirements change management [8, 9], requirements implementation [10], release planning [11,12] and requirements management [13].

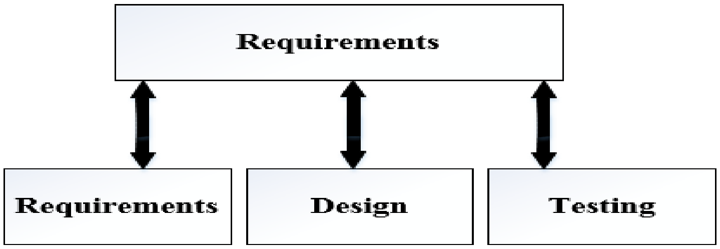


Figure 2. Software requirements traceability

Several studies have tried to propose relationships between software requirements for several purposes. For example, Karlsson et al. [14] created a preliminary set of requirements dependency types in order to use it in requirements selection such as prioritizing task of software requirements. Furthermore, Pohl [15] and Dahlst-edt and Persson [16] used literature reviews particularly in the requirements engineering field to propose several dependency types among software requirements. Carlshamre et al. [17] studied the effect of requirements dependencies in release planning process by organizing a particular industry survey on requirements. In addition, Goknil, Kurtev, and van den Berg [18] investigated the applicability of using dependency types between requirements for analyzing the process of impact change.

Requirements traceability [19] has two main modes, inter-requirements traceability and extra-requirements traceability. Inter-requirements traceability focuses on tracing requirements with other requirements while extra-requirements traceability focuses on tracing requirements with other software artifacts (i.e., source code, test cases, and etc.). Inter-requirements traceability influences a number of activities during the software development process: requirements implementation, consistency checking, and change impact. The scope of our research considers the inter-requirements traceability rather than the extra-requirements traceability because we intend to investigate the relationship types between security and functional requirements.

1.3. Security and Functional Requirements

Security requirements are a specific class of non-functional requirements that focuses on system confidentiality, reliability, integrity, and availability that required to be fulfilled in order to attain the intended security features for the system. Meaning that, it is not investigated until all of the functional requirements have been fully identified. Security requirements describe how the system should follow a reliable procedure to protect itself from information disclosure, information threat, and information corruption. It guarantees that the system applies a certain manner to keep a secure access, secure communication, and secure storage. While functional requirements describe the system behavior by expressing it as inputs to the system, outputs from the system, and relationships between inputs and outputs. There are several parties involved in the requirements engineering process, each has different backgrounds, knowledge, concerns, perceptions, and expressions. These parties include customers, users, domain experts, requirements engineers, software developers, and etc., they might have a conflicting viewpoints [20] among them in some cases. As a result, there is an essential necessity to discover relationships between security and functional requirements to detect such conflict, show which security requirements affect which functional requirements, and determine the type for this effect.

1.4. Domain Ontology

Domain ontology is a very popular semantic processing technique that classifies concepts and relations between concepts within a particular domain. Due to the difficulty of automated analysis of software requirements, and the bottlenecks in dealing with Natural Language Processing (NLP) techniques, semantic techniques are required to analyze software requirements in higher quality rather than traditional natural language techniques.

Using domain ontology in requirements engineering helps to better understanding and capturing the requirements information. Additionally, reusing and sharing of concepts and relations that are represented by the ontology. An example of a domain ontology is an ontology for the health domain. In order to apply domain ontology for analyzing software requirements, several studies used domain ontology for many goals. For example, Kaiya and Saeki [21] used domain ontology to measure the overall quality of requirements document. Also, Li et al. [22] used domain ontology to simplify the requirements elicitation process.

Assawamekin, Sunetnanta, and Pluempitiwiriyawejet [23] used a domain ontology to solve semantic dissimilarities between multi-views requirements. In addition, Jyothilakshmi and Samuel [24] used domain ontology to extract the class diagram from the functional requirements. In another study, Lee et al. [25] used domain ontology models to extract the main attributes and constraints of regulatory requirements.

Moreover, L´opez, Cysneiros, and Astudillo [26] used ontology to describe non-functional requirements and design foundation by using a goal dependency graph in order to reuse the knowledge about non-functional requirements. Also, Falbo, Guizzardi, and Duarte [27] presented ontology approach for domain engineering that linked ontologies and objects in object oriented methodology.

Jureta, Mylopoulos, and Faulkner [28] proposed an essential ontology for software requirements engineering that covered the main concerns and intentions of the stakeholders to make a nearly complete view about requirements engineering problem. Our approach uses domain ontology for automating the traceability between security and functional requirements.

1.5. Rule Based System

IF-THEN [29] rules are one of the most common forms of the knowledge representation. Systems that employed such rules as the major representation are called rule based systems. The statements after IF are called the conditions (i.e., antecedents), those after THEN are called the conclusions (i.e., consequents). We decide to use a rule based system to construct new detection rules because it has the following advantages: Very useful approach to represent knowledge, simple to construct, and easy to understand without extra interpretation.

1.6. Problem Statement

Manual traceability requires an extreme amount of effort and time. As a result, automating of requirements traceability among various types of software requirements is very necessary to make the task of requirements traceability more achievable and cost effective. Several requirements tractability tools proposed in the literature, but many lack to use a predefined deterministic rules to identify relationships between requirements. Moreover, several approaches [30, 31, 32, 33] that were proposed to identify cooperative, conflicting, and irrelevant relationships however these current approaches have several limitations such as: limited to fuzzy requirements, a little consideration to show the effects of security requirements upon functional requirements, and lack of capturing the syntactic and semantic aspects of requirements. To overcome such problems, we developed a requirements diagram, Security Functional Requirements Diagram (SFRD) that shows the relationships types between security and functional requirements. Our approach is designed to overcome the current limitations by capturing syntactic and semantic aspects of requirements, shows the effects of security requirements onto functional requirements, and it is applicable for both certain and uncertain (i.e., fuzzy) requirements.

Our solution is a hybrid approach, it uses syntactic parsing technique to extract requirements statements constructs, domain ontology to create a knowledge repository about security and functional requirements domain, and a rule based system to build multiple detection rules that identify the effect types of security requirements upon functional requirements. In addition, we develop a tool called Detection Rules Constructor (DRC) that efficiently applies the domain ontology based tractability approach and graphically generate SFRD.

Our traceability tool captures the syntactic and semantic aspects of the natural requirements text and applies 343 different detection rules to identify three types of effect: (1) cooperative, (2) conflicting, and (3) irrelevant.

CHAPTER 2. LITERATURE REVIEW

This research is related to security and functional requirements, inter-requirements relationships, and traceability tools. This chapter presents a background material for these areas.

2.1. Security and Functional Requirements

There are many studies that defined precisely security and functional requirements. For example, Haley et al. [34] described the security requirements as a part of non-functional requirements that constrained the functional requirements of a system. Kotonya and Sommerville [35] defined security requirements as restrictions or constraints on system services. Also, Rushby [36] defined security requirements as mostly concerns of the system that must not occur.

While functional requirements [37] 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. Another study [38] defined it as what the system should do in terms of its functionalities and services that it should provide. Security requirements have several types [39] such as the following:

1. Identification Requirements: a security requirement that identifies the level to which an application shall identify the externals users.

    - Example: The system shall identify all of its users before allowing them to use its resources.

2. Authentication Requirements: a security requirement that identifies the level to which the system shall verify the identity of its externals users.

- Example: The system shall verify the identity of all of its users.

3. Authorization Requirements: a security requirement that identifies the access privileges of authenticated users.

    - Example: The system shall allow each user to gain access to all his/her account information.

4. Immunity Requirements: a security requirement that identifies the level to which the system shall protect itself from infection such as viruses, hackers, and worms.

    - Example: The system shall protect itself from viruses.

5. Integrity Requirements: a security requirement that identifies the level to which keeps the information safe from illegal corruption or modification.

    - Example: The system shall prevent the unauthorized corruption of all user information.

6. Privacy Requirements: a security requirement that identifies the level to which the system shall keep all critical data private from illegal users.

    - Example: The system shall not allow unauthorized users access to any stored data.

Since security requirements [34, 35] are considered as constraints on functional requirements of the system. The investigation of relationships between security and functional requirements helps for checking consistency, prioritizing security requirements, finding association between security requirements, and analyzing the impact of relationships change.

2.2. Cooperative, Conflicting, and Irrelevant Relationships

Identifying cooperative, conflicting, and irrelevant relationships between software requirements is very important because it affects several significant factors such as requirements implementation, consistency checking, and impact change. In the literature, several studies had used different approaches to identify cooperative, conflicting, and irrelevant relationships.

Egyed and Grünbacher [30] proposed an approach for determining conflicts and cooperation dependencies among software requirements using the quality attributes of requirements and automated trace analyzer technique. Their approach consists of the following steps. First, manually classifying of software requirements into related quality attributes using international taxonomy of requirements. Second, automatic detection of cooperation and conflicts based on their associated quality characteristics using a specific model of potential impacts among software requirements, as well as ISO 9126 standardization. Third, automatic generation of dependencies among requirements by using a trace analyzer tool for the associated source code. A trace analyzer uses testing to generate trace dependencies. Fourth, filtering all the related attributes for both cooperation and conflicts among software requirements by measuring the requirements overlapping. For example, if two requirements execute the same lines of code, then they are overlapping, this reflects that there exists a trace dependency between them, and they can be used for detecting conflict and cooperation. The no-overlapping case means that there is no trace dependencies between them and they cannot be used for detecting conflict and cooperation effects since they are irrelevant.

This approach has several limitations, it is only applicable to product requirements and cannot be applied to the other process requirements like schedule and budget requirements, the need for source code with the associating requirements. In addition, does not capture the syntactic and semantic sides of software requirements.

Liu [31] identified the conflicts and cooperation dependencies among uncertain software requirements by using fuzzy logic technique. He represented each requirement (R) as a satisfaction function (i.e., fuzzy subset of the requirement domain D). For example, if a satisfaction of the first requirement increases the satisfaction of the second requirement, then they are cooperative while if satisfaction of the first requirement decreases the satisfaction of the second requirement, then they are conflicting. Also, they are irrelevant if the satisfaction of the first requirement does not have any effect on the satisfaction of the second requirement. The disadvantages for this approach are: it is only applicable to the fuzzy requirements that have uncertain terms like low, high, and medium. Also, it ignores the semantic and syntactic aspects of software requirements.

In another study, Temponi, Yen, and Tiao [32] employed Quality Functional Deployment (QFD) methodology to translate customer satisfaction (i.e., customer requirements) into organization functions (i.e., technical requirements) and they used fuzzy logic based requirements analysis to represent QFD because it can handle the imprecise expressions in the requirements. Based on this representation, they identified cooperative and conflicting relationships between customer and technical requirements. In addition, based on the existing customer and technical requirements relationships. This approach considers only the imprecise customer and technical requirements, it does not show impact of non-functional requirements onto functional requirements.

Lee and Xue [33] used a goal based approach to analyze and evaluate the cooperative, conflicting, and irrelevant relationships between user requirements. They represented the user requirements by building the use case models with the associated goals. In order to find the user requirements relationships, they found the relationships between use cases and the associated goals by identifying the certain satisfied and denied use cases for each goal, then finding interactions between goals in both use case and system level. For example, at the use case level, if a use case satisfied with the first goal and denied with the second goal, then both goals have a conflicting relationship while if a use case satisfied with the first and second goals, then both goals have a cooperative relationship. At the system level, both goals are conflicting if the cooperative predicate is false and conflicting predicate is true, and they are cooperative if cooperative predicate is true and conflicting predicate is false while they are irrelevant if both of cooperative and conflicting predicate is false. This approach considers only the relationships between functional requirements (i.e., user requirements) and does not find the relationships between functional and non-functional requirements. This approach applies in particular to the user requirements. Our approach differs from the current approaches. Our approach covers both non-functional and functional requirements rather than just focusing on one type of requirements like fuzzy requirements. Also, it captures both syntactic and semantic aspects of software requirements as in Table 1. It is a hybrid approach that applies: (1) syntactic parsing technique for requirements statements to decompose each requirement statement into three constructs (i.e., Subject-concept, Verb-concept, and Complement-concept), (2) domain ontology to capture the semantic meaning for security and functional requirements concepts and relations, and (3) rule based system to construct detection rules that identify cooperative, conflicting, and irrelevant relationships.

Table 1. Features matrix for the requirements traceability approaches

Features	Requirements traceability approach				
	Egyed and Grünbacher [30]	Liu [31]	Temponi, Yen, and Tiao [32]	Lee and Xue [33]	Our approach
Capturing syntactic and semantic aspects of requirements	✗	✗	✗	✗	✓
Finding relationships between non-functional and functional requirements	✗	✗	✓	✗	✓
Applying for fuzzy requirements	✗	✓	✓	✗	✓
Applying for precise requirements	✓	✗	✗	✓	✓

2.3. Requirements Traceability Tools

Automated solutions to requirements traceability face a difficult challenge due to the need to handle multiple types of software requirements and relationships that generated during the requirements analysis. Additionally, these software requirements are typically written using natural language making them very challenging to process automatically. Furthermore, the task of searching the requirements documents manually looking for the relationships is time and effort consuming. This situation increases the necessity to automate the requirements traceability among various types of software requirements to make the task of requirements traceability more achievable and cost effective. Automating the requirements traceability has several benefits such: reduce the effort of manual finding of relationships, and reduce the human errors that appears when trying to produce this information manually. Several requirements traceability tools have been identified to relate requirements with other requirements, or requirements with other software artifacts.

However, less attention has been paid for relating functional requirements with other non-functional requirements, and there is a lack of using a predefined deterministic rules to identify relationships between requirements.

For instance, Rational RequisitePro [40] provides only two general relations types between requirements: traceFrom and traceTo but there is no specific definitions for these relations. It used to manage requirements and allows the user to associate attributes and rational with requirement documents. However, the user manually adds relationships if other software artifacts are involved. Also, SysML [41] has three relations: contain, copy, and derive. However, there are no precise rules on how to determine these relations.

In addition, Telelogic Doors [42], linking requirements to design items, test plans, test cases and other requirements but there is no deterministic types for the requirements relations such the user can specify his own relation type, they define specific relationship types but they do not attach any meaning to these types. TopTeam Analyst [43], there are four relation types: trace, tracing links into, impact, and used in. None of these relation types have formal semantic definition. Also, QUARCC [44] supports finding dependencies between non-functional requirements, and does not consider dependencies between functional and non-functional requirements. These tools have lack to support consistency checking of the relations.

Our tool addresses the previous tools limitations such as: (1) defines several user-predefined detection rules for each type of relations (i.e., cooperative, conflicting, and irrelevant), (2) relates functional and non-functional requirements, (3) captures syntactic and semantic aspects of requirements, and (4) checks the consistency of the relations.

CHAPTER 3. THE PROPOSED APPROACH

3.1. Domain Ontology Based Detection Approach

To decrease the ambiguity and inconsistency of the informal natural language of requirements, our approach captures both the syntactic and semantic features of requirements statements as in Figure 3. The syntactic aspect of requirements focuses on the grammatical analysis of requirements statements parts like Subject, Verb, and Complement, and the semantic aspect of requirements focuses on understanding the meaning of requirements. In our research, we integrate syntactic analysis (i.e., syntactic parsing) and semantic analysis (i.e., domain ontology), which helps to identify relationships between security and functional requirements.

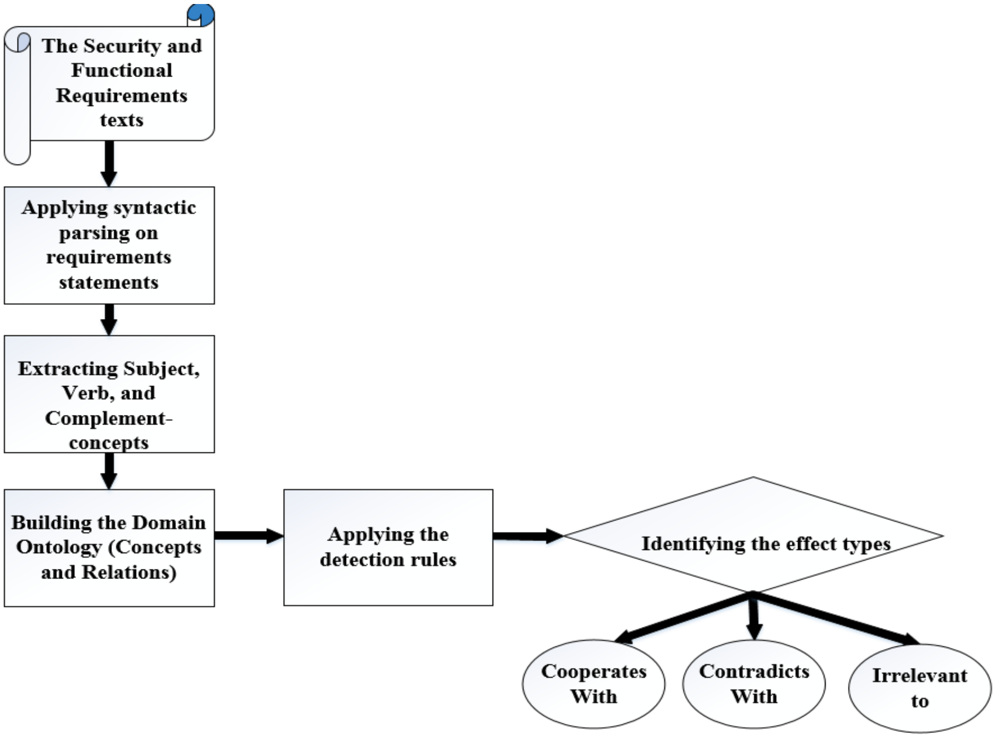

The Security and Functional Requirements texts
Applying syntactic parsing on requirements statements
Extracting Subject, Verb, and Complement-concepts
Building the Domain Ontology (Concepts and Relations)
Applying the detection rules
Identifying the effect types
Cooperates With
Contradicts With
Irrelevant to

Figure 3. An overview of our approach

Domain ontology is a very common semantic processing technique that is used to analyze the requirements specifications. Thus, we used it to determine the relations between the security and functional requirements concepts. Also, we decided to use a rule based system to construct our new detection rules to determine the types of relationships. Our approach includes the following steps:

1. Applying syntactic parsing on the security and functional requirements statements by using an online parsing tool [45] to split each security and functional requirements statements into Subject, Verb, and Complement constructs. We consider each requirement construct as a single concept, each requirement statement has three concepts: Subject, Verb, and Complement. Each concept can be either a single term or a phrase.
2. Building the domain ontology to represent the domain concepts and relations for a particular domain. Security and functional requirement Concepts have three types: (1) Subject-concept (which represents requirement entity), (2) Verb-concept (which represents requirement action) and (3) Complement concept (which represents the extra description for the requirement entity or the Object that represents requirement target). The relations have five main types: Generalization, Aggregation, Association, Synonyms, and Antonyms. The domain ontology contains two sets: (1) Concepts = {Subject-concept, Verb-concept, and Complement-concept} and (2) Relations between concepts = {Generalization, Aggregation, Association, Synonyms, and Antonyms}. Figure 4 shows our domain ontology, in which the first three relations (i.e., Generalization, Aggregation, and Association) are extracted from the built class diagram. Also, the last two relations (i.e., Synonyms and Antonyms) are extracted from WordNet [46], it is a lexical database of English, and then mapped into the domain ontology as in Figure 4.

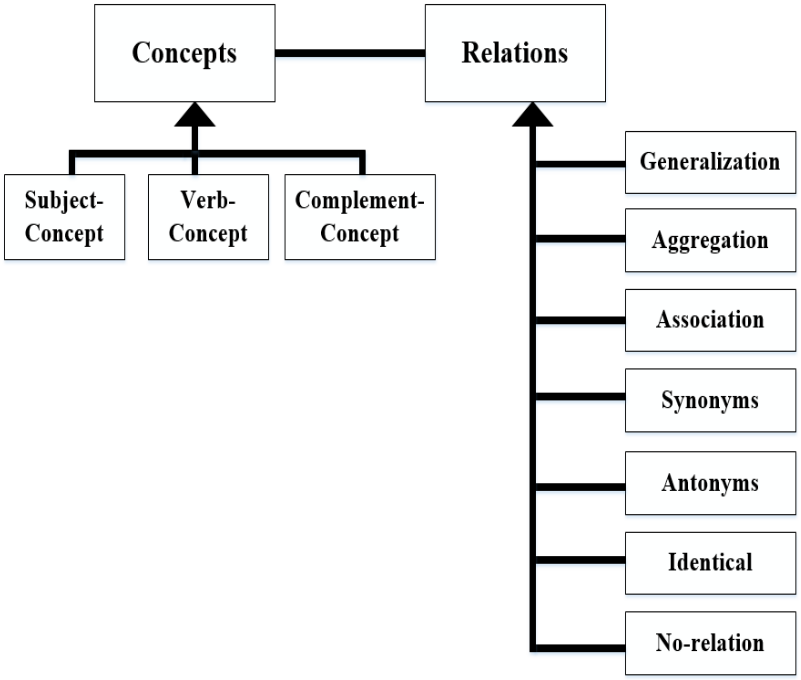


Figure 4. Domain ontology for our approach

3. 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 a reasoning engine and the rule based system has been considered as a reasoning engine for our domain ontology.
4. Identifying the Effect-type of the security requirement onto the functional requirement based on the 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 tracing model for both security and functional requirements while SFTM is a requirements diagram that illustrates the effects of security requirements upon functional requirements.

3.2. Security Functional Tracing Model (SFTM)

Inter-requirements traceability refers to finding the relationships between requirements. Identifying cooperative, conflicting, and irrelevant relationships between security and functional requirements is very important requirements engineering activity because it affects several significant software activities such as requirements implementation, consistency checking, and impact of requirements change.

In this research, we propose a tracing model called SFTM that applies a whole tracing to cover the relationships between the whole set of security and functional requirements. Our automated tracing model is a quicker method and relatively inexpensive rather than manual tracing approach.

This model consists of four essential sub sets: Security requirements, Functional requirements, Tracing pairs, and Effect types. Each set is a subset of the superset SFTM. For example, if we have system x, we represent the main components of our tracing model as follows:

1. Security Requirements set, denoted as: $SR_{(x)} = \{SR_i, SR_{i+1,}\ldots, SR_n\}$, where x is the name of software system.

2. Functional Requirements set, denoted as: $FR_{(x)} = \{FR_j, FR_{j+1}, \ldots, FR_m\}$.

3. Tracing Pairs set, denoted as $T_{(x)} = \{ (SR_i, FR_j),(SR_i, FR_{j+1}),\ldots,(SR_i, FR_m), (SR_{i+1}, FR_j),(SR_{i+1}, FR_{j+1}),\ldots,(SR_{i+1}, FR_m),\ldots, (SR_n, FR_j),(SR_n, FR_{j+1}),\ldots,(SR_n, FR_m)\}$.

4. Effect types set, denoted as $E_{(x)}$ = {Cooperates with, Conflicts with, and Irrelevant to}, here we read the effect type starting from functional requirement side then effect type then security requirement, for instance: FR_i “Cooperates with” with SR_i.

SFTM is a super-set that contains the four previous subsets and is represented as: SFTM (x) = {SR(x), FR(x), T(x), E(x)}. The idea behind this approach is to trace the first security requirement (SRi) with the whole set of functional requirements {FRj, FRj+1,…,FRm}. Then, for each single tracing pair (SRi, FRj) there will be a single particular effect type by using the detection rules. The same tracing technique will be applied respectively for the second security requirement (SRi+1) and similarly continue until the last security requirement (SRn).For example, if there is system A that has three security requirements {SR1, SR2, SR3} and 6 functional requirements {FR1,…, FR6}. The tracing approach is to map the first security requirement (SR1) with all 6 functional requirements {FR1,…, FR6}. Next, the second security requirement (SR2), until the third security requirement (SR3) proceeds through the tracing of 6 functional requirements. For each pair, the relationship type (i.e., effect type) will be identified based on the predefined detection rules.

As a result, in total there are 18 tracing pairs with 18 effect types (i.e., 6 tracing links for each security requirement and 6 effect types for each security requirement). Figure 3 shows the tracing process and the associated effect types for SR1 with all 6 functional requirements {FR1,…, FR6}. In Figure 5, there are 6 tracing links between SR1 and the whole set of functional requirements {FR1,…, FR6} as well as 6 effect types.

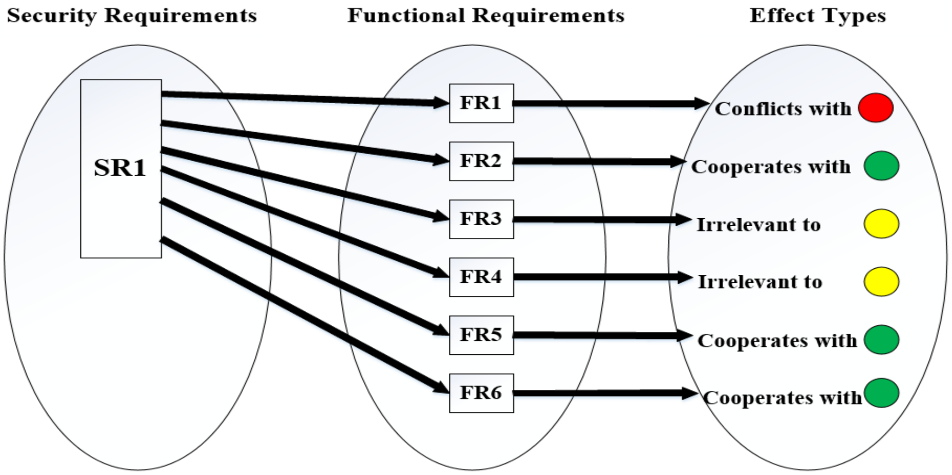


Figure 5. Security functional tracing of SR1 for system A

Figure 6 shows the tracing process and the associated effect types for SR2 with all 6 functional requirements {FR1,…, FR6}. There are 6 tracing links and 6 effect types such as: Conflicts with, Cooperates with, Irrelevant to, Cooperates with, Cooperates with, Cooperates with.

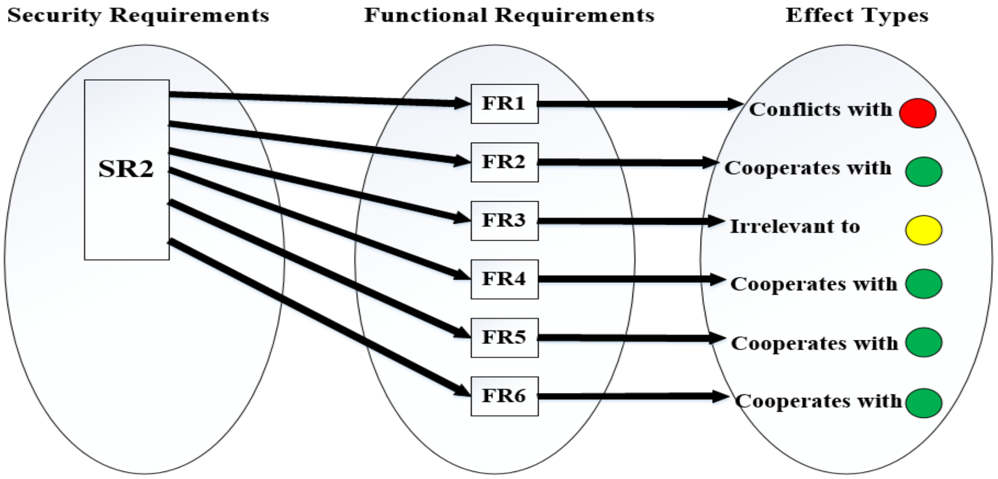


Figure 6. Security functional tracing of SR2 for system A

Figure 7 shows the tracing process and the associated effect types for SR3 with all 6 functional requirements {FR1,…, FR6}. There are 6 tracing links and 6 effect types such as: Irrelevant to, Irrelevant to, Conflicts with, Conflicts with, Cooperates with, Irrelevant to.

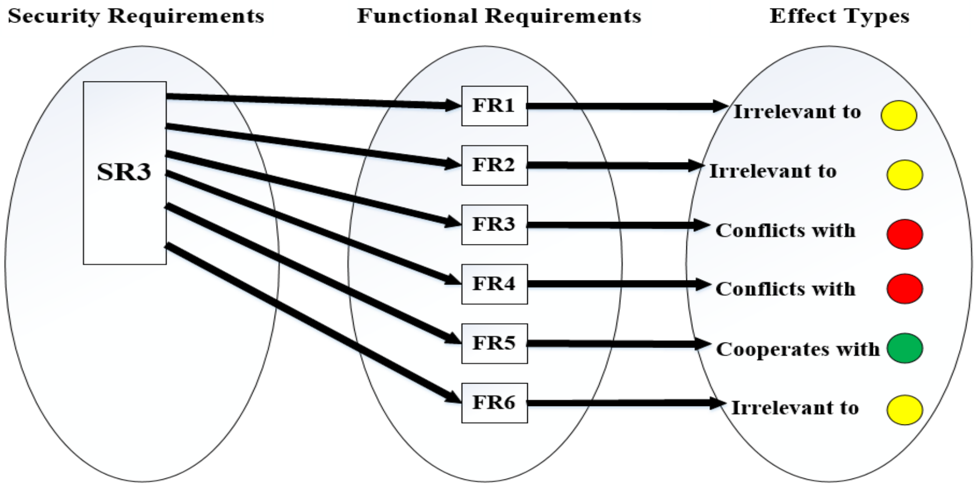


Figure 7. Security functional tracing of SR3 for system A

Based on our traced model, system A will be represented as the following mathematical notations:

1. SR$_{\text{(system A)}}$ ={ SR1, SR2, SR3}

2. FR$_{\text{(system A)}}$ = { FR1,… ,FR6}

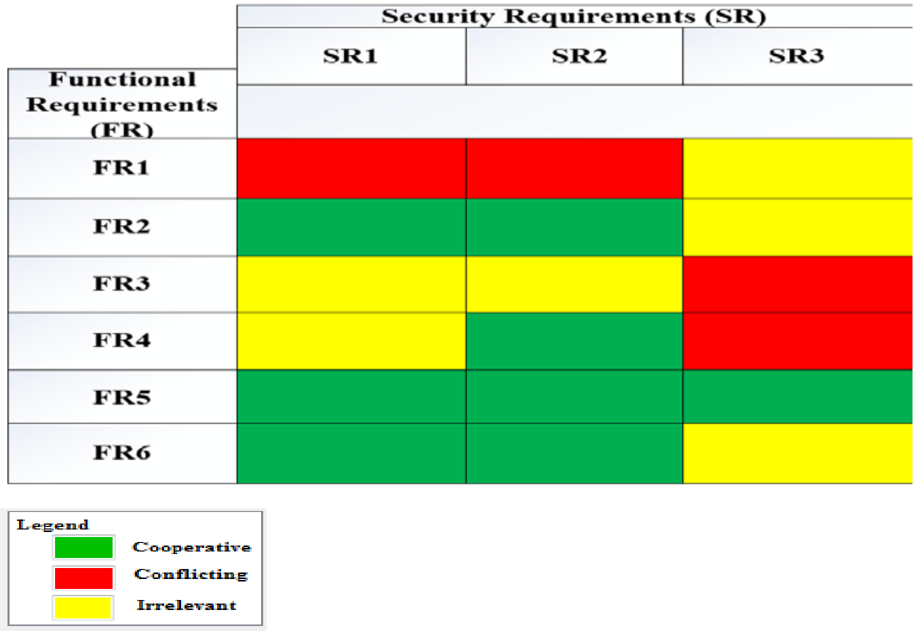
3. T $_{\text{(system A)}}$ = {(SR1, FR1),…, (SR1, FR6),…, (SR3, FR1),..., (SR3, FR6)}.

4. E$_{\text{(system A)}}$ = { Conflicts with,…,Cooperates with,…, Irrelevant to,…, Irrelevant to }

5. SFTM $_{\text{(system A)}}$ = {{ SR1, SR2},{ FR1,…,FR6},{(SR1, FR1),… ,(SR1, FR6),…,(SR3, FR1),…,(SR2, FR6)}, {Conflicts with,…,Cooperates with,…, Irrelevant to,…, Irrelevant to}}

3.3. Security Functional Requirement Diagram (SFRD)

Since security requirements are considered as constraints on functional requirements, we called relationships types as effects types. We propose requirements diagram called Security Functional Requirement Diagram (SFRD) that shows the effect types of security requirements upon Functional requirements. The diagram has been graphed as a tabular form, in which the columns represent the security requirements while the rows represent the functional requirements. Each cell (i.e. intersection between each column and each row) indicates whether the functional requirement is affected/or not by the security requirement. Different types of effects (i.e., relationships) have been indicated by different colors. For example, a functional requirement that cooperates with security requirement is shown in green, a functional requirement that conflicts with security requirement is shown in red, and a functional requirement that is irrelevant to security requirement is shown in yellow. Consequently, SFRD for System A will be tabulated as shown in Figure 8.



	Security Requirements (SR)		
	SR1	SR2	SR3
Functional Requirements (FR)			
FR1	🟥	🟥	🟨
FR2	🟩	🟩	🟨
FR3	🟨	🟨	🟥
FR4	🟨	🟩	🟥
FR5	🟩	🟩	🟩
FR6	🟩	🟩	🟨

Legend

🟩	Cooperative
🟥	Conflicting
🟨	Irrelevant

Figure 8. SFRD for system A

3.4. Domain Ontology for Security and Functional Requirements

In respect to the traceability goal, our traceability approach uses domain ontology to represent domain knowledge for security and functional requirements. The domain ontology has the concepts and the relations, the concepts have three types:

1. Subject-concept (i.e., requirement entity)
2. Verb-concept (i.e., requirement action)
3. Complement-concept (i.e., extra description for the requirement entity, or the object that represents requirement target). For example, in this security requirement statement "The system shall not contain any failure". "The system" represents the Subject-concept, "shall not contain" represents the Verb-concept, and "any failure" represents the Complement-concept.

For the relations between concepts, we used the class diagram to find Generalization, Aggregation, and Association relations [47]. In addition, we consider if two concepts have a pre-condition and post-condition relation, then they have an association. For example, register is a pre-condition for log in functionality. Furthermore, authenticate is a post-condition for log in functionality.

Additionally, we use the WordNet to find Synonyms and Antonyms relations [48]. Also, we add two more relations between concepts that reflect "Exact matching" and "No-matching" cases. Each of these relations is defined as follows:

1. Generalization: Generalization (also called inheritance) occurs when one concept (child) inherits some properties from another concept (parent). For example, an audio player is a music player.

2. Aggregation: Aggregation occurs when one concept is part of another concept (whole). For example, an engine is a part a car.

3. Association: Association represents a binary interaction between two concepts. For example, the instructor grades the course.

4. Synonyms: A Synonym is defined as two different concepts that have similar meanings. For example, Close and Shut are Synonyms.

5. Antonyms: An Antonym is when two concepts that have opposite meanings. For example, Add and Remove are Antonyms.

6. Identical: An Identical relation occurs when both concepts have the same name (i.e., exact matching). For example, Identical (personal information, personal information).

7. No-relation: No-relation occurs when both concepts do not have any of the previous specified relations. For example, No-relation (user password, customer address).

We build the domain ontology, as a class diagram [49], which includes a single class for each concept. Subject-concept, Verb-concept, and Complement-concept are stereotyped as <<Subject>>, <<Verb>>, and <<Compl>> respectively. A concept can be a single term or a whole phrase. In our approach, during the parsing step we handle positive and negative modifiers that usually used with the verbs in the natural language of the requirements description, the positive modifiers indicated for the synonyms while the negative modifiers indicated for the antonyms. Table 2 shows some of the common modifiers that we capture in our traceability approach.

Table 2. Negative and positive modifiers for inferring the implicit relations

Modifier	Type	Relation meaning	Example
Except	Negative	Antonyms	read, except read
Must	Positive	Synonyms	protect, must protect
Never	Negative	Antonyms	never modify, modify
Shall	Positive	Synonyms	encrypt, shall encrypt
Only	Negative	Antonyms	only read, write
Shall Not	Negative	Antonyms	access, shall not access
Prevent	Negative	Antonyms	display, prevent display

In addition, during the concepts extraction, we adjust only the user requirement statement (i.e., if requirement statement has both system and user as a Subject-concept) while the system requirement statement remain without any change (i.e., if requirement statement has only system as a Subject-concept) such as in Table 3.

Table 3. Adjustments of concepts extraction for user requirements statements

Requirement statement	Subject-concept	Verb-concept	Complement-concept
The system shall not allow the user to change the personal information.	user	shall not change	personal information
The system shall allow the user to print the invoice.	user	print	invoice
The system shall enable the user to select the shipping method.	user	select	shipping method
The system shall let the user to change the payment option.	user	change	payment option
The system shall list all the available promotions.	system	list	available promotions

3.5. The Definitions for Effect Types

Discovering effect types (i.e., relationship types) between security and functional requirements has several advantages: improving the understandability of security and functional requirements implementation, prioritizing security requirements, finding association between security requirements, and detecting inconsistency for security requirements:

1. Cooperative effect: the functional requirement is positively affected by the security requirement, so both security and functional requirements can be implemented at the same time.

    - Example: The system shall encrypt the user profile information (SR)

      The system shall allow user to enter his password (FR)

2. Conflicting effect: the functional requirement is negatively affected by the security requirement, so both security and functional requirements cannot be implemented at the same time.

    - Example: The user shall not access the student’s academic records (SR)

      The user can modify the student’s grades (FR)

3. Irrelevant effect: the functional requirement is neither positively nor negatively affected by the security requirement, so the implementation of the security requirement does not affect the implementation of the functional requirement.

    - Example: The system shall allow user to only read the payment history (SR)

      The user shall select the product category (FR)

3.6. The Detection Rules

Based on the resulting domain concepts and relations that we obtained from domain ontology, we constructed the detection rules to identify the effect types between security and functional requirements. These detection rules are constructed using IF-THEN rules. Every detection rule has three relations and one effect type as in Figure 9.

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* ∈ {Cooperative, Conflicting, Irrelevant}

Figure 9. The structure of the detection rule

The first relation relates two Subject-concepts, the second relation relates two Verb-concepts, and the third relation relates two Complement-concepts. Each detection rule has only one effect type. In addition, each effect type has several detection rules that have been constructed to identify it.

3.7. Detection Rules Constructor Tool Support

Requirements traceability [50] is mostly acknowledged to be a highly manual and difficult to process. Moreover, automated solutions to requirements traceability considered to be a very challenge problem due to the following reasons: 1) the need to handle several types of requirements and relations that generated during the requirements analysis, 2) ambiguous requirements frequently raise the cost of software projects, and 3) requirements have been written using natural language that makes them very hard to process automatically.

Most requirements traceability tools have paid less consideration for relating functional requirements with other non-functional requirements, and there is a lack of using a predefined deterministic rules to identify relationships between these requirements. As a result we developed a requirements traceability tool called DRC using C#.Net, it mainly uses to let the user identify the effect types of security requirements upon functional requirements, and it manipulates the three main parts of our traceability approach: domain ontology, effect types, and detection rules.

Our tool efficiently automates the SFRM and finally generates the SFRD. By using the DRC, the user can generate three outputs files: 1) matching parsing requirements with domain ontology, 2) matching tracing security and functional requirements with the associated detection rules numbers, and 3) generating SFRD that graphically showing security requirements effects on functional requirements with labeling the associated detection rules numbers for each effect. Figure 10 shows the domain ontology of our tool, it combines the three types of domain concepts and seven types of the domain relations.

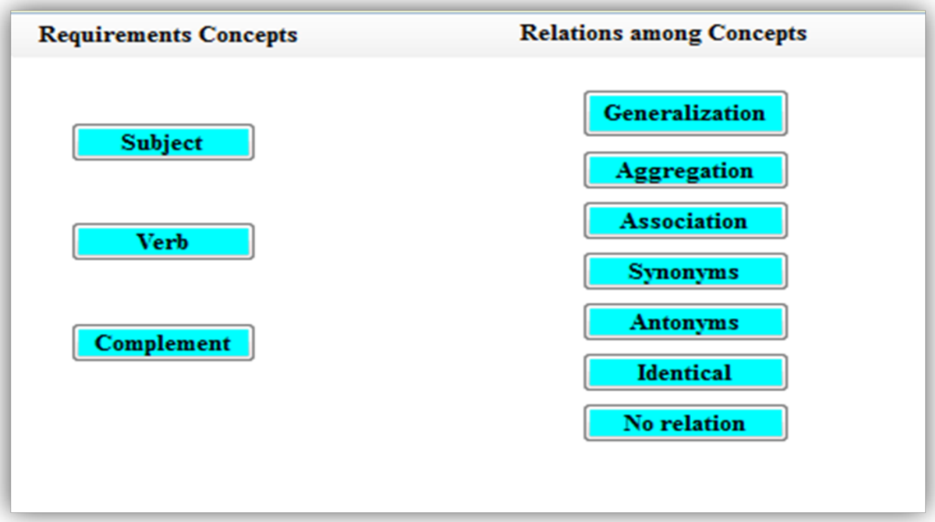


Figure 10. The domain ontology interface

The domain ontology interface let user to create a knowledge repository of concepts and relations for security and functional requirements domain, the tool will automatically generate a tree view for the all selected concepts and relations as in Figure 11.

A tree view is a graphical control element that presents a hierarchical view of domain ontology information. Each main node represents requirement concept, and sub nodes represent relations between concepts.

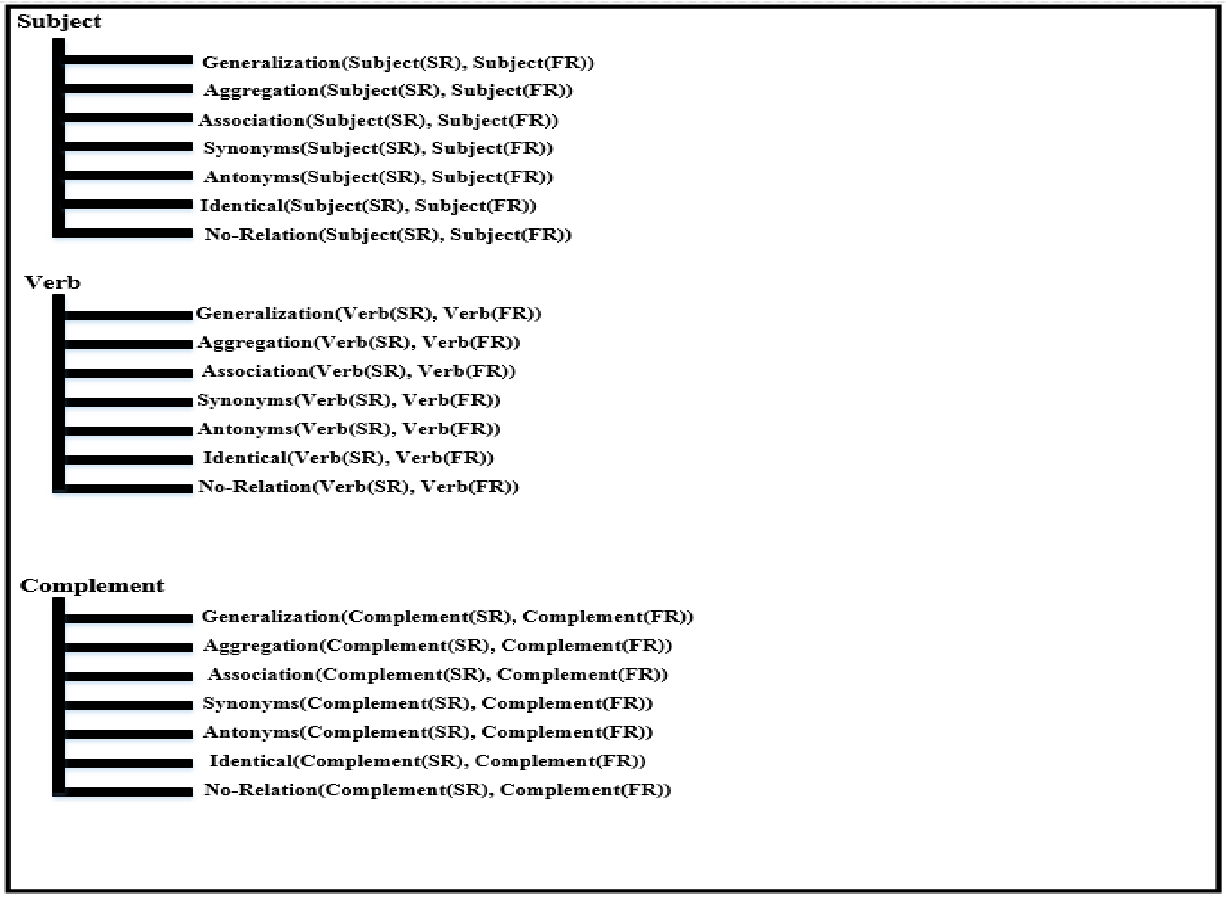
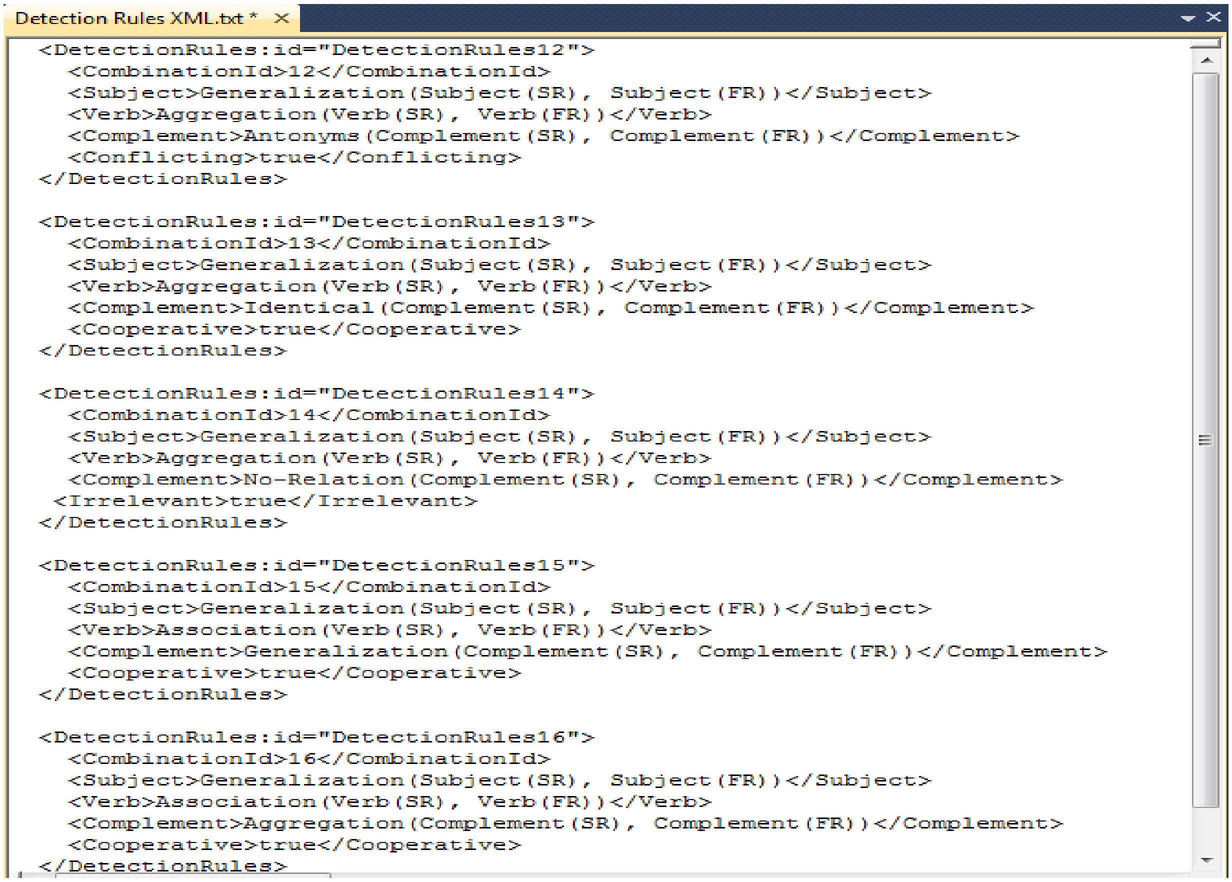

Subject
Generalization(Subject(SR), Subject(FR))
Aggregation(Subject(SR), Subject(FR))
Association(Subject(SR), Subject(FR))
Synonyms(Subject(SR), Subject(FR))
Antonyms(Subject(SR), Subject(FR))
Identical(Subject(SR), Subject(FR))
No-Relation(Subject(SR), Subject(FR))
Verb
Generalization(Verb(SR), Verb(FR))
Aggregation(Verb(SR), Verb(FR))
Association(Verb(SR), Verb(FR))
Synonyms(Verb(SR), Verb(FR))
Antonyms(Verb(SR), Verb(FR))
Identical(Verb(SR), Verb(FR))
No-Relation(Verb(SR), Verb(FR))
Complement
Generalization(Complement(SR), Complement(FR))
Aggregation(Complement(SR), Complement(FR))
Association(Complement(SR), Complement(FR))
Synonyms(Complement(SR), Complement(FR))
Antonyms(Complement(SR), Complement(FR))
Identical(Complement(SR), Complement(FR))
No-Relation(Complement(SR), Complement(FR))

Figure 11. Tree view for security and functional requirements concepts and relations

In our traceability approach, we have three main concepts and seven relations for each of these concepts, as a result we will have 343 possible combinations for the detection rules.

After the detection rules have been constructed, the user can save all of the detection rules and the associated effect types into an XML file. The detection rules that used in our approach are expressed in XML as well as the relation discovery is based on a rule-based system. The detection of the effects starts by converting all relationships between security and functional requirements into XML file as in Figure 12.



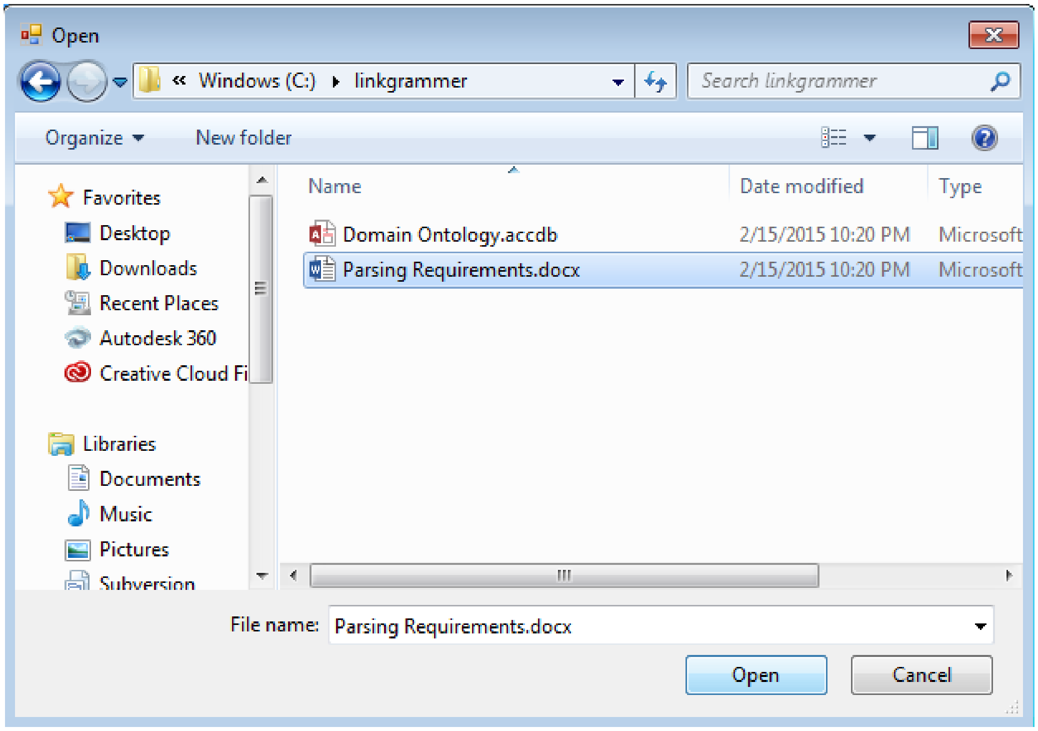
```
  <DetectionRules:id="DetectionRules12">
    <CombinationId>12</CombinationId>
    <Subject>Generalization(Subject(SR), Subject(FR))</Subject>
    <Verb>Aggregation(Verb(SR), Verb(FR))</Verb>
    <Complement>Antonyms(Complement(SR), Complement(FR))</Complement>
    <Conflicting>true</Conflicting>
  </DetectionRules>

  <DetectionRules:id="DetectionRules13">
    <CombinationId>13</CombinationId>
    <Subject>Generalization(Subject(SR), Subject(FR))</Subject>
    <Verb>Aggregation(Verb(SR), Verb(FR))</Verb>
    <Complement>Identical(Complement(SR), Complement(FR))</Complement>
    <Cooperative>true</Cooperative>
  </DetectionRules>

  <DetectionRules:id="DetectionRules14">
    <CombinationId>14</CombinationId>
    <Subject>Generalization(Subject(SR), Subject(FR))</Subject>
    <Verb>Aggregation(Verb(SR), Verb(FR))</Verb>
    <Complement>No-Relation(Complement(SR), Complement(FR))</Complement>
   <Irrelevant>true</Irrelevant>
  </DetectionRules>

  <DetectionRules:id="DetectionRules15">
    <CombinationId>15</CombinationId>
    <Subject>Generalization(Subject(SR), Subject(FR))</Subject>
    <Verb>Association(Verb(SR), Verb(FR))</Verb>
    <Complement>Generalization(Complement(SR), Complement(FR))</Complement>
    <Cooperative>true</Cooperative>
  </DetectionRules>

  <DetectionRules:id="DetectionRules16">
    <CombinationId>16</CombinationId>
    <Subject>Generalization(Subject(SR), Subject(FR))</Subject>
    <Verb>Association(Verb(SR), Verb(FR))</Verb>
    <Complement>Aggregation(Complement(SR), Complement(FR))</Complement>
    <Cooperative>true</Cooperative>
  </DetectionRules>
```

Figure 12. A portion of the detection rules XML file

Our approach identifies 131 detection rules for cooperative effect, 92 detection rules for conflicting effect, and 120 different detection rules for irrelevant effect. Overall, DRC handles 343 distinct detection rules that cover the all possible combinations.

3.7.1. Inputs for DRC Tool

Since our approach combines the syntactic and semantic aspects of the requirements, then our tool takes two inputs files: 1) Microsoft Word file which represents parsing security and functional requirements, and 2) Microsoft Access file which represents the stored domain ontology as in Figure 13.



Name	Date modified	Type
Domain Ontology.accdb	2/15/2015 10:20 PM	Microsoft
Parsing Requirements.docx	2/15/2015 10:20 PM	Microsoft

File name: Parsing Requirements.docx

Figure 13. Inputs files for DRC

The MS Word file has the requirements statements with the associated Subject-concept, Verb-concept, and Complement-concept. Each requirement has been labeled to indicate the type and the number for each requirement, and the MS Access file has the information repository for security and functional requirements.

3.7.2. Outputs for DRC Tool

Our tool handles 343 different possible combinations for the detection rules, Figure 14 shows the main interface of our tool that allows the user to automatically generate three outputs files based on reading the two inputs files. In this section we will show a brief description about each output.

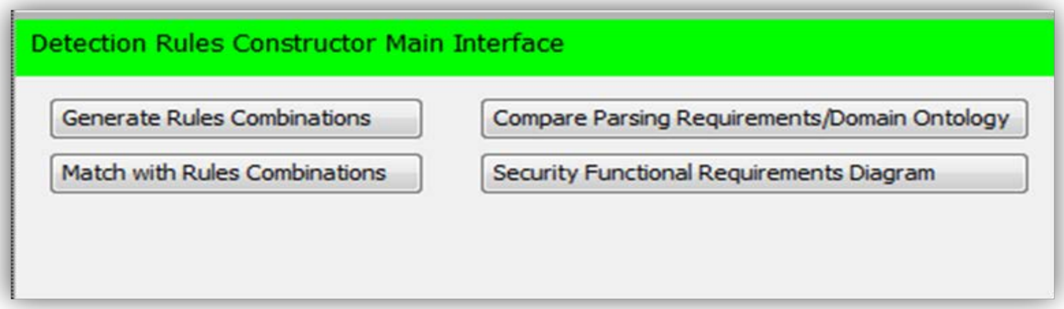

Detection Rules Constructor Main Interface
Generate Rules Combinations
Compare Parsing Requirements/Domain Ontology
Match with Rules Combinations
Security Functional Requirements Diagram

Figure 14. Main interface for DRC

A. Comparing parsing security and functional requirements with the domain ontology: This is the first output file from our tool such when the user click onto compare task on the main interface. The input dialog will prompt the user to select the parsing requirements Word file. Then, DRC will automatically apply the comparison with the built domain ontology, and generate the whole tracing pairs of security and functional requirements with the corresponding Subject-Relation, Verb-Relation, and Complement-Relation.

B. Matching security and functional requirements with the associated detection rules-ID: This is the second output file from the tool, it matches the whole tracing security and functional requirements with the associated general detection rules numbers. It is the same as the first output file plus adding a new column for the associated detection rules numbers.

C. Generating Security and Functional Requirements Diagram: This is the third output file from the tool. Based on the first and second output files, our tool generates SFRD which significantly provides a well-defined structured manner for illustrating requirements traceability. Each cell in SFRD is labeled with the associated detection rule number that identifies a certain type of effect. SFRD increases the user understandability for inferring potential relationships between security and functional requirements.

3.8. Importance of Investigating Effect Types of Security Requirements upon Functional Requirements

3.8.1. Prioritization of Security Requirements

Prioritizing requirements based on importance and dependency lead to have many benefits [51, 52]. Firstly, it can be used to show the features in each software product version or release planning [53]. Secondly, prioritization supports an accurate method for choosing the most important security requirements to implement, which may result in a reduction a number of implemented functional requirements [54].

As a result, one of the significant benefits of our approach is to prioritize security requirements based on counting number of functional requirements that cooperate with security requirement (i.e. Weight). Each security requirement has a different number of associated dependent functional requirements. System A has three security requirements (SR1, SR2, and SR3) as in Figure 15. 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) as follows:

- Weight (SR1) = # of cooperated functional requirements nodes with SR1= 3
- Weight (SR2) = # of cooperated functional requirements nodes with SR2=4
- Weight (SR3) = # of cooperated functional requirements nodes with SR3= 1

Therefore, SR2 has the highest priority, and SR3 has the lowest priority. The highest priority security requirement positively affects many functional requirements more than any other security requirements. This prioritization helps requirements engineer in the requirements selection process.

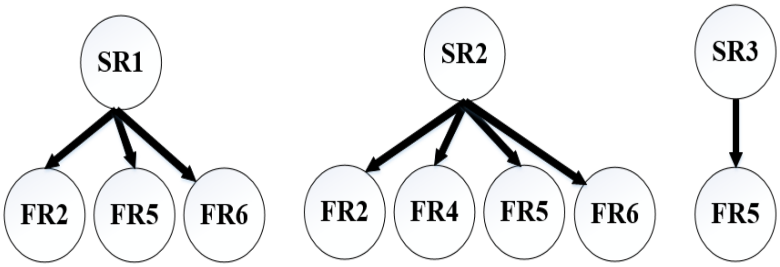


Figure 15. The cooperated functional requirements nodes for system A

3.8.2. Finding Association among Security Requirements

In our approach, we create an association graph for security requirements by using Jaccard similarity technique [55]. We measure the association between the security requirements nodes based on the mutual cooperated functional requirements nodes. The technique is defined as the quotient between the intersection and the union between the security requirements as in Equation 1.

$$\text{Association}(\text{SRi}, \text{SRj}) = \frac{|\text{SRi} \cap \text{SRj}|}{|\text{SRi} \cup \text{SRj}|} \qquad \text{(Eq. 1)}$$

- Where (SRi, SRj) represents security requirements.

- $SRi \cap SRj$: represents the number of common cooperated functional requirements nodes between SR i and SR j.

- $SRi \cup SRj$: represents the number of all cooperated functional requirements nodes in both SR i and SR j.

We use an Association to measure the similarity values between the dependent security requirements. For instance, if Association value is equal to zero, both security requirements are dissimilar while if Association is equal to one both security requirements are identical. By applying Jaccard similarity for the example in Figure 16, the resulting association values have been calculated as follows:

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

The association between SR1 and SR2 is 0.75, which reflects strongly dependent requirements (i.e., highly connected requirements) while the association between SR2 and SR3 is 0.25, which reflects weakly dependent requirements (i.e., lowly connected requirements).

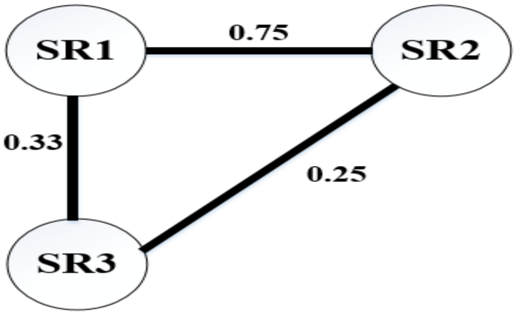


Figure 16. A weighted association graph for system A

3.8.3. Finding Inconsistency between Security and Functional Requirements

The additional valuable benefit that we can get from using SFRD is to find the inconsistency [56] for each single security requirement. The inconsistency ratio is calculated by considering all the conflicting effects relative to the total number of effects as in Equation 2.

$$\text{Inconsistency Ratio (SRi)} = \frac{\text{Number of confliting effect for SRi}}{\text{Total number of all effect types for SRi}} \qquad \text{(Eq. 2)}$$

- Where, total number of all effects = number of cooperative effect + number of conflicting effect + number of irrelevant effect. For system A, we can find the following information:
- Number of conflicting effect for SR1= 1
- Number of conflicting effect for SR2= 1
- Number of conflicting effect for SR3= 2
- Total number of all effects=6, then the inconsistency ratios for the security requirements as follows:
- Inconsistency ratio (SR1) = 1/6 = 0.16
- Inconsistency ratio (SR2) = 1/6 = 0.16
- Inconsistency ratio (SR3) = 2/6 = 0.33

SR3 has the highest inconsistency ratio among all the security requirements for system A, while both SR1 and SR2 have the same ratio. As a result, we can find that if a security requirement has more conflicting effects, this leads to increase the inconsistency for the requirements. Our goal is to detect the inconsistency between security and functional requirements in order to increase the clarity as well as reduce the confusion that will appear in the requirement analysis of the software development life cycle.

CHAPTER 4. EXPERIMENTAL EVALUATION

A Software requirements specification [57] is a description of the developed behavior system includes functional requirements and non-functional requirements. Functional requirements describe all the interactions the users will have with the software and non-functional requirements are requirements which impose constraints on the design or implementation, such as performance requirements, quality standards, or design constraints.

Functional requirements capture the intended behavior of the system. This behavior had been expressed as services, or functions the system is required to perform. Security requirements are non-functional requirements that define constraints and restrictions on the functional requirements that include authorization, access control, authentication, privacy, integrity, identification etc.

In this chapter, we demonstrate the effectiveness of our tool in the context of various systems that covers health, commercial, educational, and hospitality industry domains. We consider seven different requirements specifications, and for each system we show domain ontology, SFRD, and some samples for the detection rules that have been used in these systems. Also, to show the major benefits of our approach: (1) prioritizing for security requirements, (2) finding association between security requirements, and (3) determining the inconsistency ratio for security requirements.

4.1. Online Medical Database System

In order to capture the health domain, we select to apply our traceability approach to the online medical database system which aims to save time span through providing a searchable database of all past medic records.

This system facilitates the management and medical staff through using electronic database instead of the manual one. It handles the current and historical illness information for the patient, nurse information, and patient history information such as family history as well as social work history. This system provides an interesting challenge because it is semantically rich with different domain relations. Moreover, the privacy and security issues are so critical to such systems. Figure 17 shows a portion for the domain ontology for online medical database system.

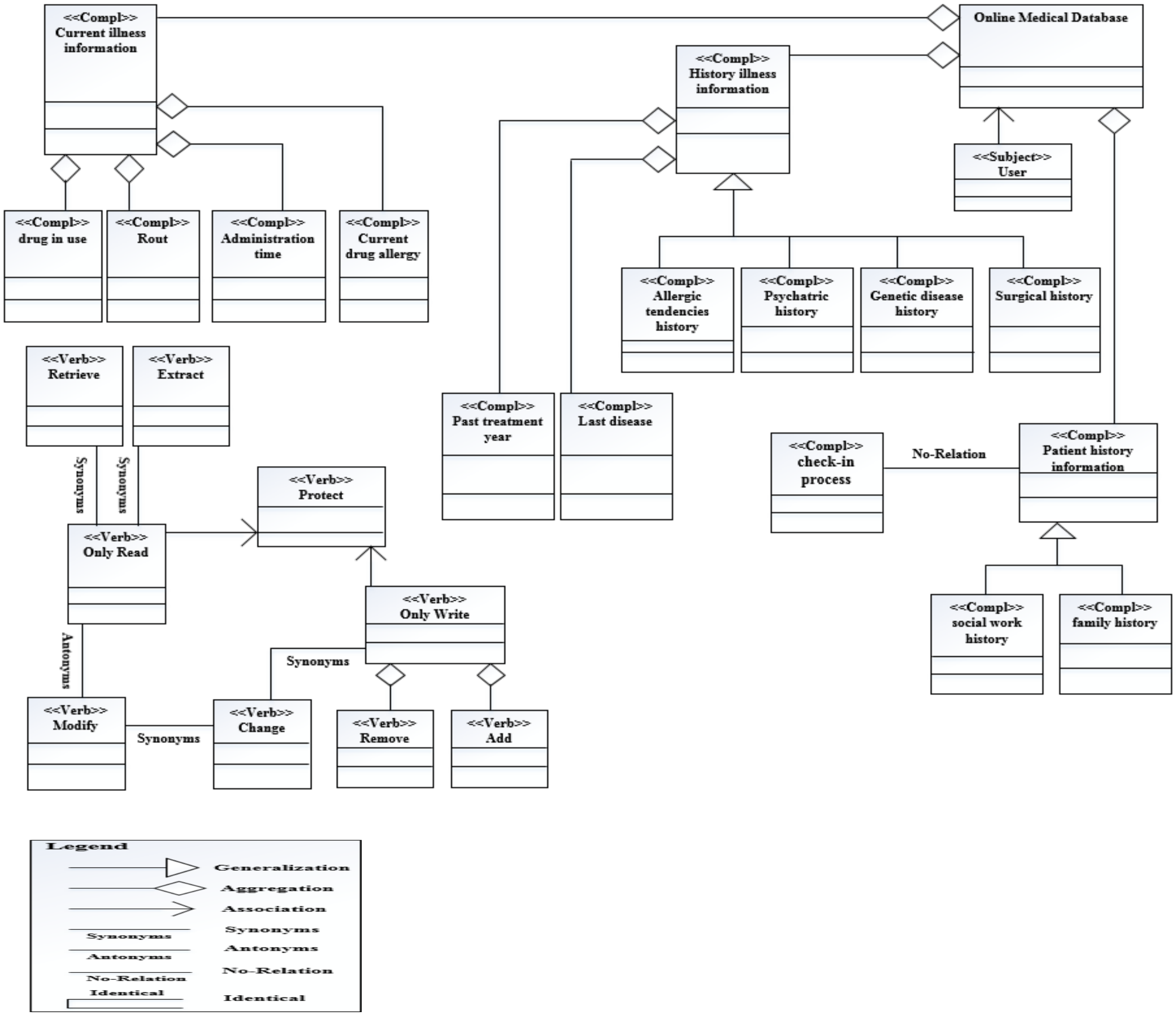
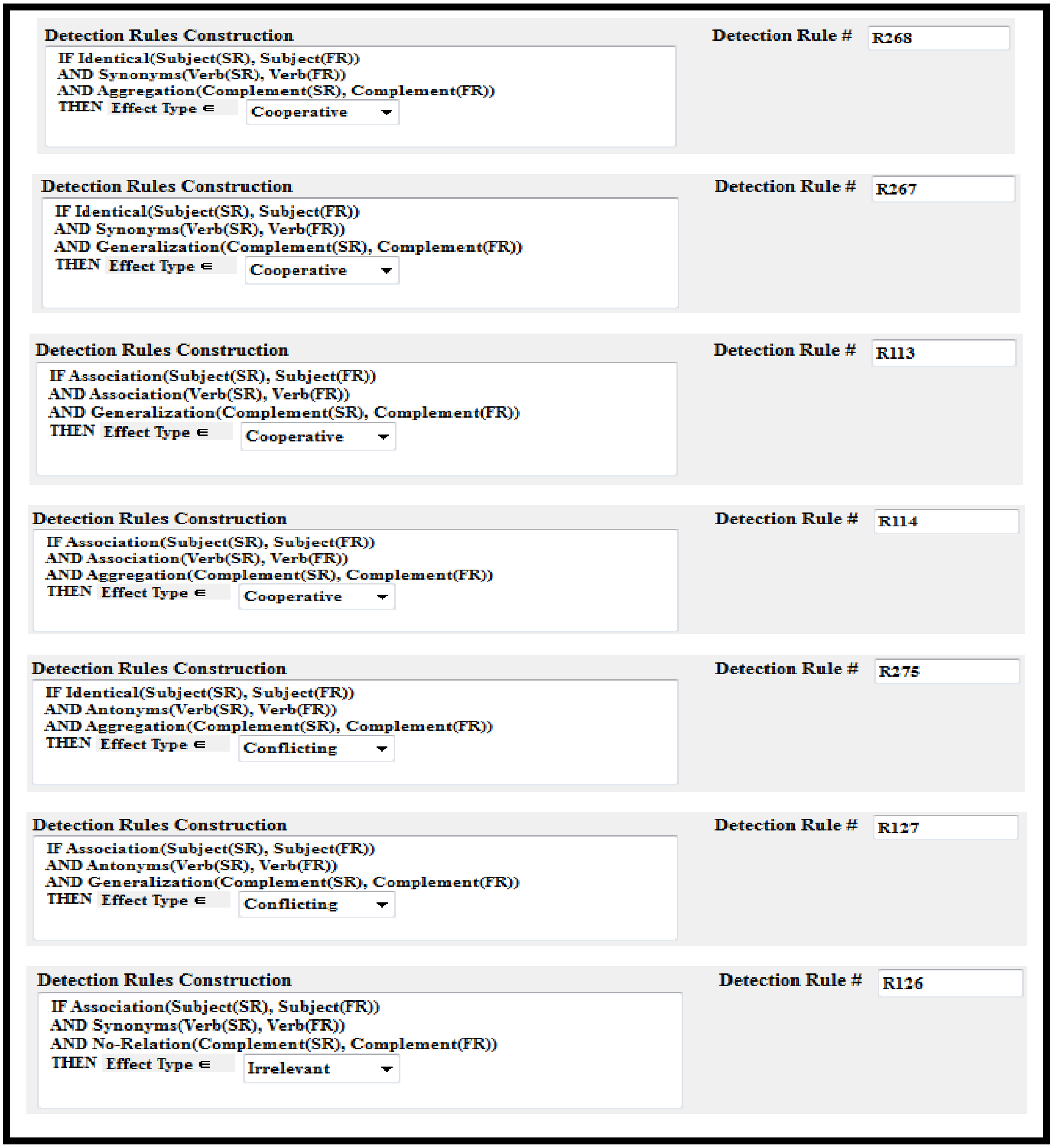


Figure 17. A part of domain ontology for an online medical database system

We use several detection rules to determine the effect types between security and functional requirements. Figure 18 shows a sample set for these rules.



Detection Rules Construction
IF Identical(Subject(SR), Subject(FR))
AND Synonyms(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative
Detection Rule # R268

Detection Rules Construction
IF Identical(Subject(SR), Subject(FR))
AND Synonyms(Verb(SR), Verb(FR))
AND Generalization(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative
Detection Rule # R267

Detection Rules Construction
IF Association(Subject(SR), Subject(FR))
AND Association(Verb(SR), Verb(FR))
AND Generalization(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative
Detection Rule # R113

Detection Rules Construction
IF Association(Subject(SR), Subject(FR))
AND Association(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative
Detection Rule # R114

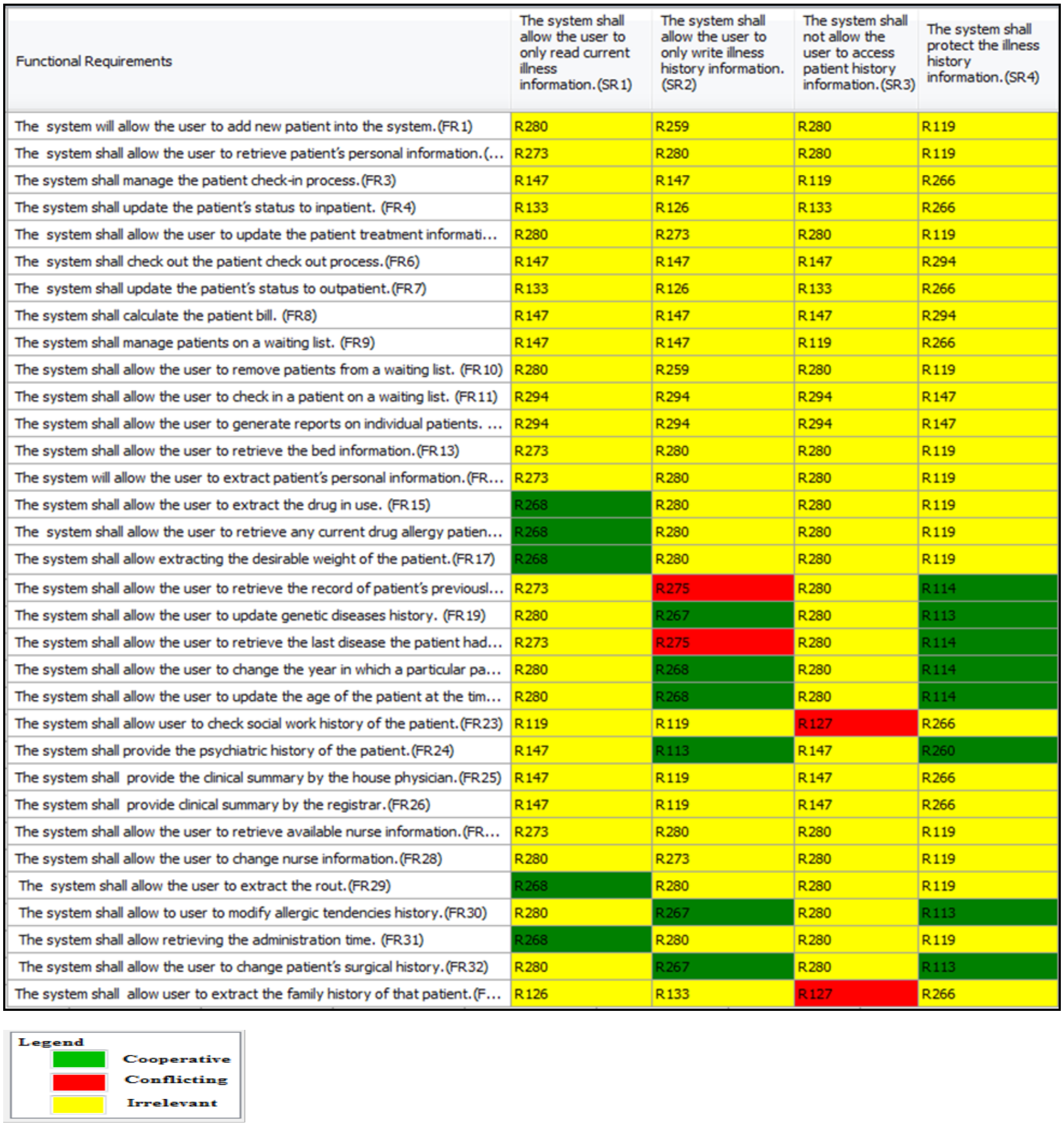
Detection Rules Construction
IF Identical(Subject(SR), Subject(FR))
AND Antonyms(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Conflicting
Detection Rule # R275

Detection Rules Construction
IF Association(Subject(SR), Subject(FR))
AND Antonyms(Verb(SR), Verb(FR))
AND Generalization(Complement(SR), Complement(FR))
THEN Effect Type ∈ Conflicting
Detection Rule # R127

Detection Rules Construction
IF Association(Subject(SR), Subject(FR))
AND Synonyms(Verb(SR), Verb(FR))
AND No-Relation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant
Detection Rule # R126

Figure 18. A sample of the detection rules for an online medical database system

In this system, DRC handles the tractability between four security requirements and 33 functional requirements, so we have 132 tracing pairs. After that, DRC automatically generate the Security Functional Requirement Diagram for an online medical database system using the stored XML detection rules. Figure 19 shows SFRD for an online medical database system.



Functional Requirements	The system shall allow the user to only read current illness information.(SR1)	The system shall allow the user to only write illness history information.(SR2)	The system shall not allow the user to access patient history information.(SR3)	The system shall protect the illness history information.(SR4)
The system will allow the user to add new patient into the system.(FR1)	R280	R259	R280	R119
The system shall allow the user to retrieve patient's personal information.(…	R273	R280	R280	R119
The system shall manage the patient check-in process.(FR3)	R147	R147	R119	R266
The system shall update the patient's status to inpatient. (FR4)	R133	R126	R133	R266
The system shall allow the user to update the patient treatment informati…	R280	R273	R280	R119
The system shall check out the patient check out process.(FR6)	R147	R147	R147	R294
The system shall update the patient's status to outpatient.(FR7)	R133	R126	R133	R266
The system shall calculate the patient bill. (FR8)	R147	R147	R147	R294
The system shall manage patients on a waiting list. (FR9)	R147	R147	R119	R266
The system shall allow the user to remove patients from a waiting list. (FR10)	R280	R259	R280	R119
The system shall allow the user to check in a patient on a waiting list. (FR11)	R294	R294	R294	R147
The system shall allow the user to generate reports on individual patients. …	R294	R294	R294	R147
The system shall allow the user to retrieve the bed information.(FR13)	R273	R280	R280	R119
The system will allow the user to extract patient's personal information.(FR…	R273	R280	R280	R119
The system shall allow the user to extract the drug in use. (FR15)	R268	R280	R280	R119
The system shall allow the user to retrieve any current drug allergy patien…	R268	R280	R280	R119
The system shall allow extracting the desirable weight of the patient.(FR17)	R268	R280	R280	R119
The system shall allow the user to retrieve the record of patient's previousl…	R273	R275	R280	R114
The system shall allow the user to update genetic diseases history. (FR19)	R280	R267	R280	R113
The system shall allow the user to retrieve the last disease the patient had…	R273	R275	R280	R114
The system shall allow the user to change the year in which a particular pa…	R280	R268	R280	R114
The system shall allow the user to update the age of the patient at the tim…	R280	R268	R280	R114
The system shall allow user to check social work history of the patient.(FR23)	R119	R119	R127	R266
The system shall provide the psychiatric history of the patient.(FR24)	R147	R113	R147	R260
The system shall provide the clinical summary by the house physician.(FR25)	R147	R119	R147	R266
The system shall provide clinical summary by the registrar.(FR26)	R147	R119	R147	R266
The system shall allow the user to retrieve available nurse information.(FR…	R273	R280	R280	R119
The system shall allow the user to change nurse information.(FR28)	R280	R273	R280	R119
The system shall allow the user to extract the rout.(FR29)	R268	R280	R280	R119
The system shall allow to user to modify allergic tendencies history.(FR30)	R280	R267	R280	R113
The system shall allow retrieving the administration time. (FR31)	R268	R280	R280	R119
The system shall allow the user to change patient's surgical history.(FR32)	R280	R267	R280	R113
The system shall allow user to extract the family history of that patient.(F…	R126	R133	R127	R266

Legend

🟩 Cooperative

🟥 Conflicting

🟨 Irrelevant

Figure 19. SFRD for an online medical database system

In Figure 19, there exists a single detection rule that can identify a single effect type and cover several requirements tracing pairs such as R114 identifies the cooperative effect for (SR4,FR18), (SR4,FR20), (SR4,FR21), and (SR4,FR22). Also, R127 identifies the conflicting effect for (SR3, FR23), and (SR2, FR33). In addition, R133 identifies the irrelevant effect for (SR1, FR4), (SR3, FR4), (SR2, FR33), (SR1, FR7), and (SR3, FR7).

1. Prioritization for security requirements: Based on SFRD, we can find clearly the cooperated functional requirements nodes for each security requirement as follows:

   - SR1 has five cooperated functional requirements: FR15, FR16, FR17, FR29, and FR31.
   - SR2 has six cooperated functional requirements: FR19, FR21, FR22, FR24, FR30, and FR32.
   - SR3 does not have any of cooperated functional requirements.
   - SR4 has eight cooperated functional requirements: FR18, FR19, FR20, FR21, FR22, FR24, FR30, and FR32. Therefore, the Weight for each security requirement is calculated as follows:
   - Weight (SR1) = 5
   - Weight (SR2) = 6
   - Weight (SR3) = 0
   - Weight (SR4) = 8, Then, prioritization for security requirements from high weight to low weight is: SR4, SR2, SR1, and SR3 respectively.

2. Finding association between security requirements: By applying Jaccard method, the obtaining association values between the security requirements of OMD system are as follows:

    - Association (SR1, SR2) = 0/11 = 0
    - Association (SR1, SR3) = 0/5 = 0
    - Association (SR1, SR4) = 0/13 = 0
    - Association (SR2, SR3) = 0/6 = 0
    - Association (SR2, SR4) = 6/8 = 0.75
    - Association (SR3, SR4) = 0/8 = 0

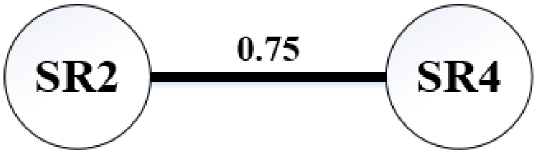


Figure 20. A weighted association graph for an online medical database system

Figure 20 shows that the association between SR2 and SR4 is 0.75, which reflects strongly associated requirements, while the other security requirements have not any association between them. These association will help the requirement engineer to figure out the strength for the dependencies among security requirements.

3. Inconsistency ratio for each security requirement: We can find the conflicting effects for each security requirement as follows:

- Number of conflicting effect for SR1 = 0
- Number of conflicting effect for SR2 = 2
- Number of conflicting effect for SR3 = 2
- Number of conflicting effect for SR4 = 0
- Total number of effects for each security requirement = 33, the inconsistency ratio is calculated by considering all the conflicting effects relative to the total number of effects as follows:
- Inconsistency ratio (SR1) = 0/33 = 0
- Inconsistency ratio (SR2) = 2/33 = 0.06
- Inconsistency ratio (SR3) = 2/33 = 0.06
- Inconsistency ratio (SR4) = 0/33 = 0

In our approach, we aim to detect the inconsistency between security and functional requirements in order to improve the understandability and reduce the natural language ambiguity for the requirements text.

Based on the inconsistency values, SR2 and SR3 have the same ratio because each of them has two conflicting effects out of 33 effects, but SR1 and SR4 have not any conflicting effect, so both have not any inconsistency.

4.2. Online Store System

In order to capture the commercial domain, we select to apply our approach to the online store system [58] which describes online sales, distribution, and marketing for the products. It also concentrates on the capabilities that required by customers and their needs while defining product features and configurations. The user can select the product from the products categorization and pay online using credit card number. Figure 21 shows a portion for the domain ontology for an online store system.

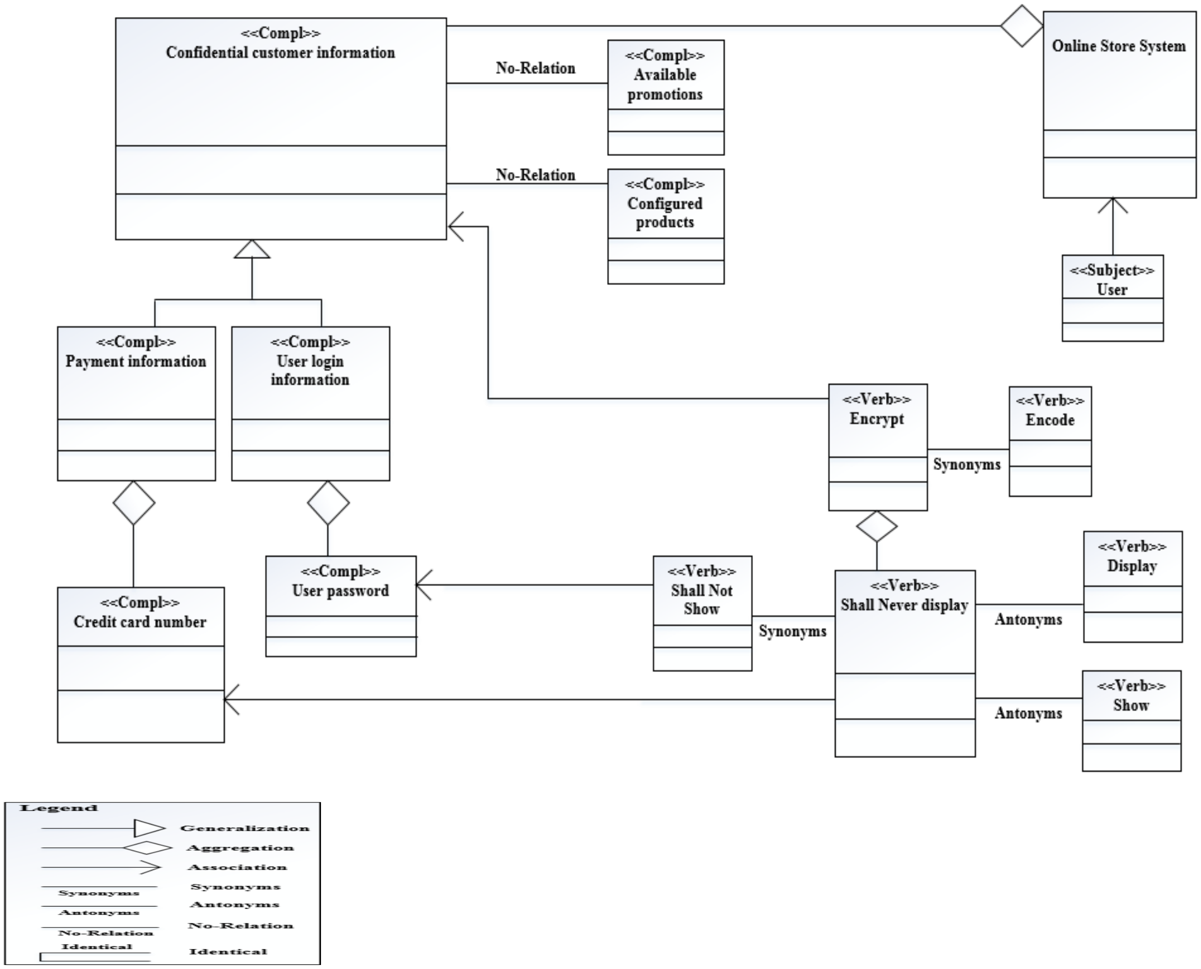
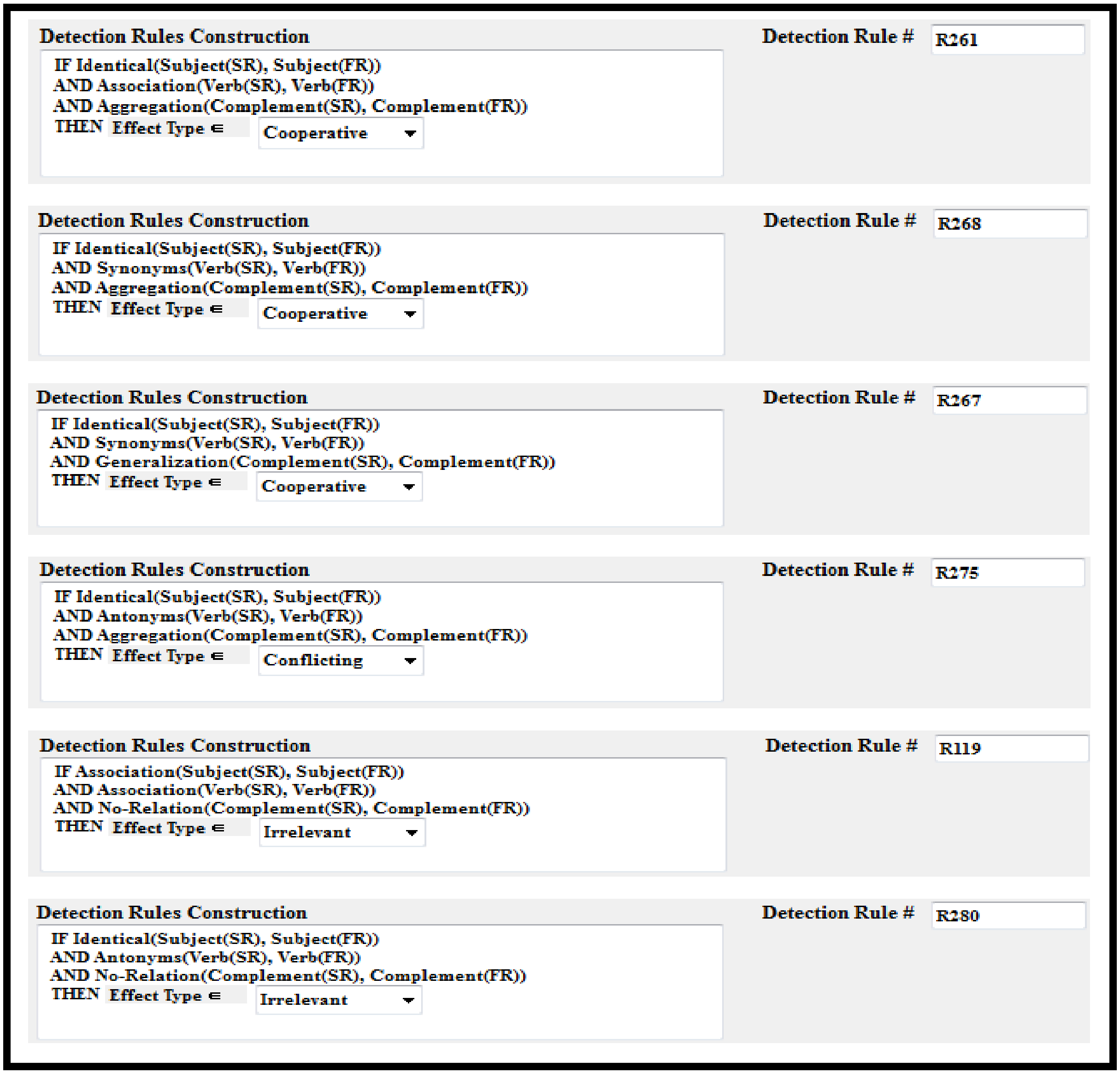


Figure 21. A part of domain ontology for an online store system

Based on the above domain ontology, confidential customer information can be either payment information or user login information (i.e., Generalization) Also, credit card number is a part of payment information (i.e., Aggregation) while user password is a part of user login information (i.e., Aggregation). Figure 22 shows a set for the detection rules that used in this system.



Detection Rules Construction

IF Identical(Subject(SR), Subject(FR))
AND Association(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rule # R261

Detection Rules Construction

IF Identical(Subject(SR), Subject(FR))
AND Synonyms(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rule # R268

Detection Rules Construction

IF Identical(Subject(SR), Subject(FR))
AND Synonyms(Verb(SR), Verb(FR))
AND Generalization(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rule # R267

Detection Rules Construction

IF Identical(Subject(SR), Subject(FR))
AND Antonyms(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Conflicting

Detection Rule # R275

Detection Rules Construction

IF Association(Subject(SR), Subject(FR))
AND Association(Verb(SR), Verb(FR))
AND No-Relation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

Detection Rule # R119

Detection Rules Construction

IF Identical(Subject(SR), Subject(FR))
AND Antonyms(Verb(SR), Verb(FR))
AND No-Relation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

Detection Rule # R280

Figure 22. A sample of the detection rules for an online store system

Here we can show an example for the first and second outputs files that we can obtain from using DRC. In the first output file, DRC finds the matching between parsing requirements text file and the domain ontology database as in Table 4, and the second output file shows the association between the parsing requirements and the matched detection rules numbers for each requirements pair as in Table 5.

Table 4. A portion of an output file for matching parsing requirements with domain ontology

Requirements Pair	**Subject-Relation**	**Verb-Relation**	**Complement-Relation**
The system shall encrypt all of the confidential customer information. (SR1), The system view detailed product categorization. (FR5)	Identical	No-Relation	No-Relation
The system shall encrypt all of the confidential customer information. (SR1), The system shall allow user to create profile and set his credential. (FR6)	Association	Association	Aggregation
The system shall encrypt all of the confidential customer information. (SR1), The system shall authenticate user credentials to view the profile. (FR7)	Identical	Association	Aggregation
The system shall encrypt all of the confidential customer information. (SR1), The system shall allow user to update the profile information. (FR8)	Association	Association	Generalization
The system shall encrypt all of the confidential customer information. (SR1), The system shall display both the active and completed order history in the customer profile. (FR9)	Identical	Association	No-Relation
The system shall never display a customer’s password. (SR2), The system shall display user login information. (FR29)	Identical	Antonyms	Aggregation
The customer’s web browser shall never display a customer credit card number. (SR3), The system shall display the credit card payment data. (FR22)	Identical	Antonyms	Aggregation

In the above table, each requirements pair has a specific combination that shows the relation types for the security and functional requirements concepts, some tracing pairs share the same combination.

Table 5. A portion of an output file for matching requirements with associated detection rules

Requirements Pair	Subject-Relation	Verb-Relation	Complement-Relation	Matched detection rules ID
The system shall encrypt all of the confidential customer information. (SR1), The system view detailed product categorization. (FR5)	Identical	No-Relation	No-Relation	294
The system shall encrypt all of the confidential customer information. (SR1), The system shall allow user to create profile and set his credential. (FR6)	Association	Association	Aggregation	114
The system shall encrypt all of the confidential customer information. (SR1), The system shall authenticate user credentials to view the profile. (FR7)	Identical	Association	Aggregation	261
The system shall encrypt all of the confidential customer information. (SR1), The system shall allow user to update the profile information. (FR8)	Association	Association	Generalization	113
The system shall encrypt all of the confidential customer information. (SR1), The system shall display both the active and completed order history in the customer profile. (FR9)	Identical	Association	No-Relation	266
The system shall never display a customer’s password. (SR2), The system shall display user login information. (FR29)	Identical	Antonyms	Aggregation	275
The customer’s web browser shall never display a customer credit card number. (SR3), system shall display credit card payment data. (FR22)	Identical	Antonyms	Aggregation	275

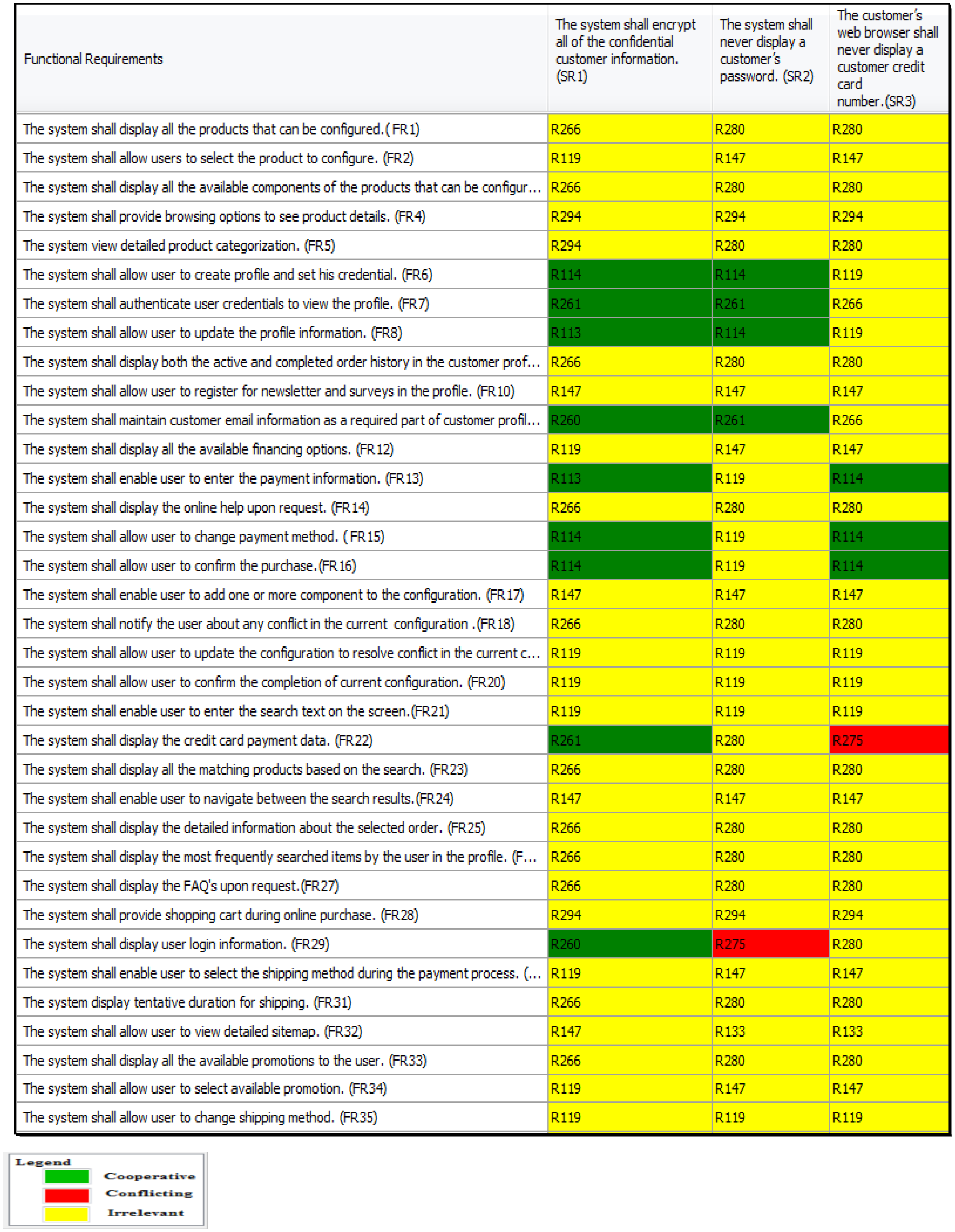
For example, (SR2, FR29), and (SR3, FR22) have the same combination: (Identical, Antonyms, Aggregation) for (Subject-Relation, Verb-Relation, Complement-relation) respectively. So, they should have the same detection rule and the same effect type.

The second output file that has been generated by the tool shows the corresponding detection rule number for each requirements pair. For example, detection rule 266 (i.e., R266) matches with the requirements pair (SR1, FR9) as in Table 5.

In the above table, the detection rule R261 identifies the cooperative effect for the tracing requirements pair (SR1, FR7), and the detection rule R294 identifies the irrelevant effect for the tracing requirements pair (SR1, FR5). Also, the detection rule R275 identifies the conflicting effect for the tracing requirements pairs (SR2, FR29), and (SR3, FR22). By using DRC, the user can generate the first table and the second table as we described above. Also, the user can generate SFRD that collects required information from the previous tables and graphically illustrates the effect type and the associated detection rule of each requirements tracing pair for an online store system as in Figure 23.

SFRD generates 105 tracing links that captures the whole tracing between the three security requirements and 35 functional requirements. Each security requirement has been traced to 35 functional requirements, and since it has three security requirements then it has in total 105 tracing links that cover all security and functional requirements sets.

In the online store system, the detection rules R114, R113, R261, and R261 identify 16 cooperative effect. Also, the detection rule R294 identifies two conflicting effect, and the detection rules R266, R119, R294, R147, R280, and R133 identify 87 irrelevant effect.



Functional Requirements	The system shall encrypt all of the confidential customer information. (SR1)	The system shall never display a customer's password. (SR2)	The customer's web browser shall never display a customer credit card number.(SR3)
The system shall display all the products that can be configured.(FR1)	🟨 R266	🟨 R280	🟨 R280
The system shall allow users to select the product to configure. (FR2)	🟨 R119	🟨 R147	🟨 R147
The system shall display all the available components of the products that can be configur…	🟨 R266	🟨 R280	🟨 R280
The system shall provide browsing options to see product details. (FR4)	🟨 R294	🟨 R294	🟨 R294
The system view detailed product categorization. (FR5)	🟨 R294	🟨 R280	🟨 R280
The system shall allow user to create profile and set his credential. (FR6)	🟩 R114	🟩 R114	🟨 R119
The system shall authenticate user credentials to view the profile. (FR7)	🟩 R261	🟩 R261	🟨 R266
The system shall allow user to update the profile information. (FR8)	🟩 R113	🟩 R114	🟨 R119
The system shall display both the active and completed order history in the customer prof…	🟨 R266	🟨 R280	🟨 R280
The system shall allow user to register for newsletter and surveys in the profile. (FR10)	🟨 R147	🟨 R147	🟨 R147
The system shall maintain customer email information as a required part of customer profil…	🟩 R260	🟩 R261	🟨 R266
The system shall display all the available financing options. (FR12)	🟨 R119	🟨 R147	🟨 R147
The system shall enable user to enter the payment information. (FR13)	🟩 R113	🟨 R119	🟩 R114
The system shall display the online help upon request. (FR14)	🟨 R266	🟨 R280	🟨 R280
The system shall allow user to change payment method. (FR15)	🟩 R114	🟨 R119	🟩 R114
The system shall allow user to confirm the purchase.(FR16)	🟩 R114	🟨 R119	🟩 R114
The system shall enable user to add one or more component to the configuration. (FR17)	🟨 R147	🟨 R147	🟨 R147
The system shall notify the user about any conflict in the current configuration .(FR18)	🟨 R266	🟨 R280	🟨 R280
The system shall allow user to update the configuration to resolve conflict in the current c…	🟨 R119	🟨 R119	🟨 R119
The system shall allow user to confirm the completion of current configuration. (FR20)	🟨 R119	🟨 R119	🟨 R119
The system shall enable user to enter the search text on the screen.(FR21)	🟨 R119	🟨 R119	🟨 R119
The system shall display the credit card payment data. (FR22)	🟩 R261	🟨 R280	🟥 R275
The system shall display all the matching products based on the search. (FR23)	🟨 R266	🟨 R280	🟨 R280
The system shall enable user to navigate between the search results.(FR24)	🟨 R147	🟨 R147	🟨 R147
The system shall display the detailed information about the selected order. (FR25)	🟨 R266	🟨 R280	🟨 R280
The system shall display the most frequently searched items by the user in the profile. (F…	🟨 R266	🟨 R280	🟨 R280
The system shall display the FAQ's upon request.(FR27)	🟨 R266	🟨 R280	🟨 R280
The system shall provide shopping cart during online purchase. (FR28)	🟨 R294	🟨 R294	🟨 R294
The system shall display user login information. (FR29)	🟩 R260	🟥 R275	🟨 R280
The system shall enable user to select the shipping method during the payment process. (…	🟨 R119	🟨 R147	🟨 R147
The system display tentative duration for shipping. (FR31)	🟨 R266	🟨 R280	🟨 R280
The system shall allow user to view detailed sitemap. (FR32)	🟨 R147	🟨 R133	🟨 R133
The system shall display all the available promotions to the user. (FR33)	🟨 R266	🟨 R280	🟨 R280
The system shall allow user to select available promotion. (FR34)	🟨 R119	🟨 R147	🟨 R147
The system shall allow user to change shipping method. (FR35)	🟨 R119	🟨 R119	🟨 R119

Legend

🟩 Cooperative
🟥 Conflicting
🟨 Irrelevant

Figure 23. SFRD for an online store system

Among all cooperative detection rules, we find that R114 identifies 8 requirements pairs, R261 identifies 4 requirements pairs, R260 identifies two requirements pairs, and R113 identifies two requirements pairs. For conflicting effect, there is only one detection rule that identifies two requirements pairs. While for irrelevant detection rules, we find that R280 identifies 22 requirements pairs, and R133 only identifies two requirements pairs.

SFRD enables the user to: (1) better understanding the requirements inter-traceability, (2) perceiving the most important security requirements, and (3) associating the similar functional requirements based on the mutual dependencies.

As we can notice from the above SFRD that all the parsing requirements pairs that have the same combination will be identified by the same effect type. Different security requirements have different effects on the same functional requirement. For example, the requirements pair (SR1, FR22) has a cooperative effect with the associated detection rule 261, (SR2, FR22) has irrelevant effect with associated detection rule 280, and the requirements pair (SR3, FR22) has conflicting effect associated detection rule 275. In addition, the requirements pair (SR1, FR29) has a cooperative effect with the associated detection rule 260, the requirements pair (SR2, FR29) has conflicting effect associated detection rule 275, and the requirements pair (SR3, FR29) has irrelevant effect with associated detection rule 280.

1. Prioritization for security requirements

One of the significant uses for the traceability 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 as follows:

- Weight (SR1) = 9
- Weight (SR2) = 4
- Weight (SR3) = 3, so the prioritization for the security requirements is: SR1, SR2, SR3.

2. Finding association between security requirements as in Figure 24:

- Association (SR1, SR2) = 4/9= 0.44
- Association (SR1, SR3) = 3/9= 0.33
- Association (SR2, SR3) = 0/7= 0.0

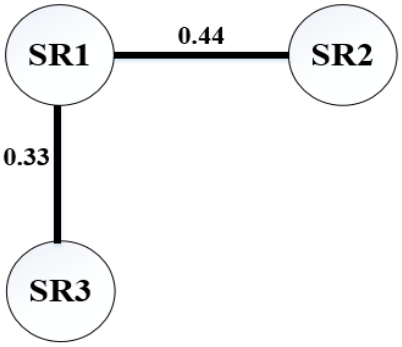


Figure 24. A weighted association graph for an online store system

3. Inconsistency ratio for each security requirement: the number of conflicting effect for each security requirement is calculated as follows:

- Number of conflicting effect for SR1 = 0
- Number of conflicting effect for SR2 = 1
- Number of conflicting effect for SR3 = 1

- Total number of effects for each security requirement = 35, then the inconsistency ratio for each security requirement is calculated as follows:
- Inconsistency ratio (SR1) = 0/35 = 0 %
- Inconsistency ratio (SR2) = 1/35 = 3%
- Inconsistency ratio (SR3) = 1/35 = 3%

4.3. Course Management System/ Students

In order to capture the educational domain, we select to apply our approach to the course management system [59], it is a software application used to control the virtual educational environments, enhance classroom education, and as a platform for distance learning programs.

This system enriched with multiple features and different strong capabilities that empower instructors to professionally manage courses contents, construct projects or assignments, and manage a collaboration environment for the students that enable the students to effectively communicate with the instructor. The course management system helps academic organizations to accomplish communication and valuation objectives.

This system specifies the requirements for students and lecturers, the functional and security requirements are separated for each of them. It has a specific glossary that defines the exact meanings for personal information, static course information, dynamic course information, and study information as in Table 6.

Table 6. Glossary of course management system requirements terms

Term	Meaning
Personal Information	Information about a person, such as name, address, a picture, interests, etc.
Study Information	Information about a person’s study progress, such as subscribed courses, grades and exam attempts.
Assistant Lecturers	Lecturers who assist the principal lecturer for a course.
Static Course Information	Information of a course which does not change while a course is given, but between semesters. This includes the lecturer, and study material.
Dynamic Course Information	Information of a course which changes while a course is given. This includes news messages, archived files and roster.
Secondary University Systems	All university systems which are shared by different departments, such as a central address book containing all kinds of personal information.
Manage	Manage involves create, set, rename, read, update, and delete.

In this system we will show how we can apply our traceability approach for tracing the security and functional requirements, the student system has four security requirements and 17 functional requirements.

The domain ontology provides a semantic repository about the concepts and relations for a particular domain. Our domain ontology captures three main concepts of a requirement statement: requirement entity, requirement action, and requirement target.

Moreover, it identifies the semantic relation between these concepts. For example, user is a subject-concept, which reflects requirement entity, and there exists a generalization relation between student and user. Figure 25 shows a portion for the domain ontology for the student requirements in the course management system.

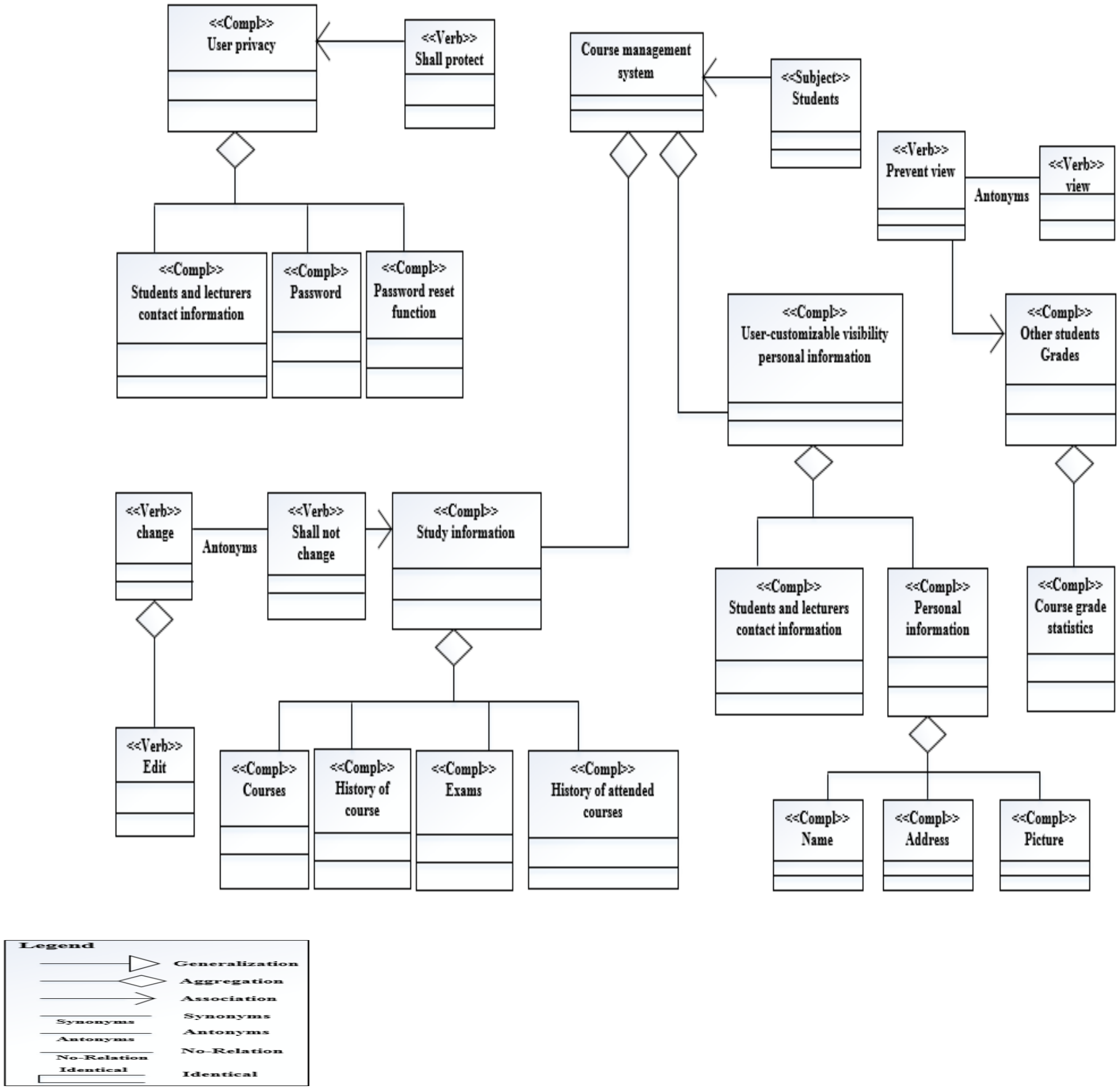


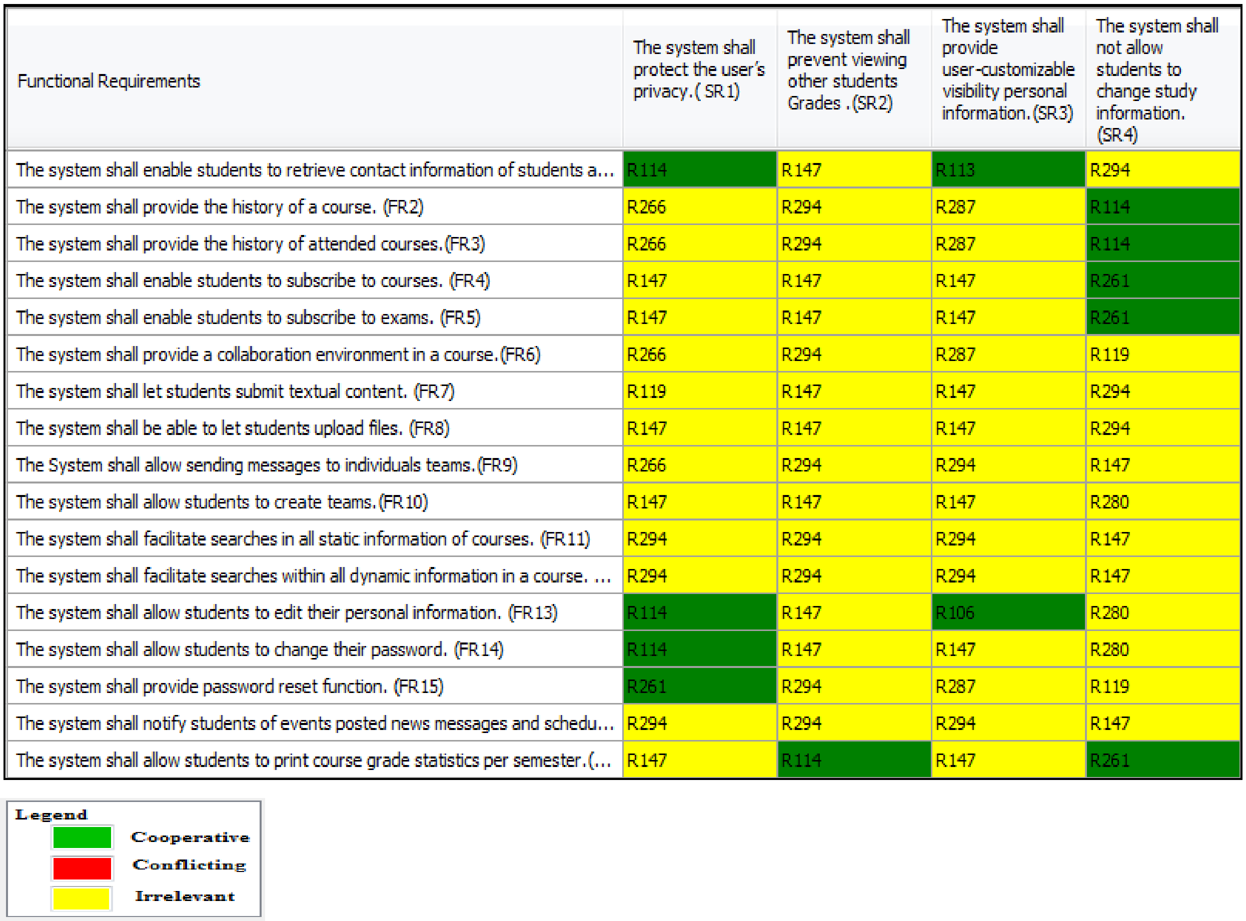
Figure 25. A part of domain ontology for students system

Also, change is a verb-concept, which reflects requirement action, and it aggregates from several actions such as: edit. Then, there exists an aggregation relation between manage and edit actions. There is antonyms relation between “change” and “shall not change”. In addition, course information is a complement-concept, which reflects requirement target, and it has two sub complement-concepts: static course information, and dynamic course information.

Then, there exists a generalization relation between them such static course information is a course information, and dynamic course information is a course information.

Similarly, user-privacy is a complement-concept, and it has many parts such: student contact information, student personal information, and student password. Also, personal information is a complement-concept and it briefly describes the information about the person and it aggregates from several complement-concepts: student name, student address, and student picture, and student interests. Also, Study information is a complement-concept and it aggregates from several complement-concepts: exams, grades, courses, and history of attended courses.

By using the domain ontology and the detection rules, our tractability tool illustrates the Security Functional Requirements Diagram for students system that has four security requirements as columns and 17 functional requirements as rows as in Figure 26. Our traceability approach simplifies the requirements traceability process and allows the user to easy understand the requirements implementation.



Functional Requirements	The system shall protect the user's privacy.(SR1)	The system shall prevent viewing other students Grades .(SR2)	The system shall provide user-customizable visibility personal information.(SR3)	The system shall not allow students to change study information. (SR4)
The system shall enable students to retrieve contact information of students a...	🟩 R114	🟨 R147	🟩 R113	🟨 R294
The system shall provide the history of a course. (FR2)	🟨 R266	🟨 R294	🟨 R287	🟩 R114
The system shall provide the history of attended courses.(FR3)	🟨 R266	🟨 R294	🟨 R287	🟩 R114
The system shall enable students to subscribe to courses. (FR4)	🟨 R147	🟨 R147	🟨 R147	🟩 R261
The system shall enable students to subscribe to exams. (FR5)	🟨 R147	🟨 R147	🟨 R147	🟩 R261
The system shall provide a collaboration environment in a course.(FR6)	🟨 R266	🟨 R294	🟨 R287	🟨 R119
The system shall let students submit textual content. (FR7)	🟨 R119	🟨 R147	🟨 R147	🟨 R294
The system shall be able to let students upload files. (FR8)	🟨 R147	🟨 R147	🟨 R147	🟨 R294
The System shall allow sending messages to individuals teams.(FR9)	🟨 R266	🟨 R294	🟨 R294	🟨 R147
The system shall allow students to create teams.(FR10)	🟨 R147	🟨 R147	🟨 R147	🟨 R280
The system shall facilitate searches in all static information of courses. (FR11)	🟨 R294	🟨 R294	🟨 R294	🟨 R147
The system shall facilitate searches within all dynamic information in a course. ...	🟨 R294	🟨 R294	🟨 R294	🟨 R147
The system shall allow students to edit their personal information. (FR13)	🟩 R114	🟨 R147	🟩 R106	🟨 R280
The system shall allow students to change their password. (FR14)	🟩 R114	🟨 R147	🟨 R147	🟨 R280
The system shall provide password reset function. (FR15)	🟩 R261	🟨 R294	🟨 R287	🟨 R119
The system shall notify students of events posted news messages and schedu...	🟨 R294	🟨 R294	🟨 R294	🟨 R147
The system shall allow students to print course grade statistics per semester.(...	🟨 R147	🟩 R114	🟨 R147	🟩 R261

Legend

🟩	Cooperative
🟥	Conflicting
🟨	Irrelevant

Figure 26. SFRD for students system

1. Prioritization for security requirements:

- Weight (SR1) = 4
- Weight (SR2) = 1
- Weight (SR3) = 2
- Weight (SR4) = 5, the prioritization from high to low is: SR4, SR1, SR3, and SR2.

2. Finding association between security requirements: We find association among security requirements as in Figure 27.

    - Association (SR1, SR2) = 0/5= 0.0
    - Association (SR1, SR3) = 2/4= 0.50
    - Association (SR1, SR4) = 0/9= 0.0
    - Association (SR2, SR3) = 0/3= 0.0
    - Association (SR2, SR4) = 1/5= 0.20
    - Association (SR3, SR4) = 0/7= 0.0

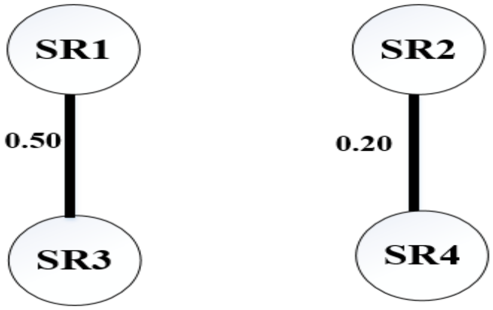


Figure 27. A weighted association graph for students system

The weighted association graph provides a very useful information that can be used requirements change process such it can be effectively used to estimate the cost of requirements change as well as considering the impact of this change. For example, in Figure 30, the association value for (SR1 and SR3) reflects that both requirements strongly connected to each other’s, while the association value for (SR2 and SR4) indicates that both requirements weakly connected.

As a result, in case of requirements changing for either SR1 or SR3, the impact of change will be higher than requirements change for SR2 or SR4. In other words, the cost of requirements change in the first case will be greater than the cost of requirements change in the second case.

3. Inconsistency: there is no inconsistency for this particular requirements set because it does not have any conflicting effect.

4.4. Course Management System/ Lecturers

This system describes the functional and security requirements for the lecturers in the course management system, we will show how we can apply our traceability approach for tracing the security and functional requirements for the lecturers.

Figure 28 shows a portion for the domain ontology for the lecturer's subsystem requirements. The domain ontology has several relations. . For example, user is a subject-concept, which reflects requirement entity, and there exists a generalization relation between lecturer and user. Also, manage is a verb-concept, which reflects requirement action, and it aggregates from several actions such as: create, insert, remove, read, update, rename, and etc. Then, there exists an aggregation relation between manage and the other actions.

In addition, dynamic course information is a complement-concept that has many complement-concepts: news messages, achieved files, roster, and course materials. Similarly, "All grades for all students" is a complement-concept and it has many complement-concepts: student's grades, team's grades, student's grading policy, and student's grade statistics. As a result, there is an aggregation relation between them. The lecturer's subsystem has three security requirements and 35 functional requirements.

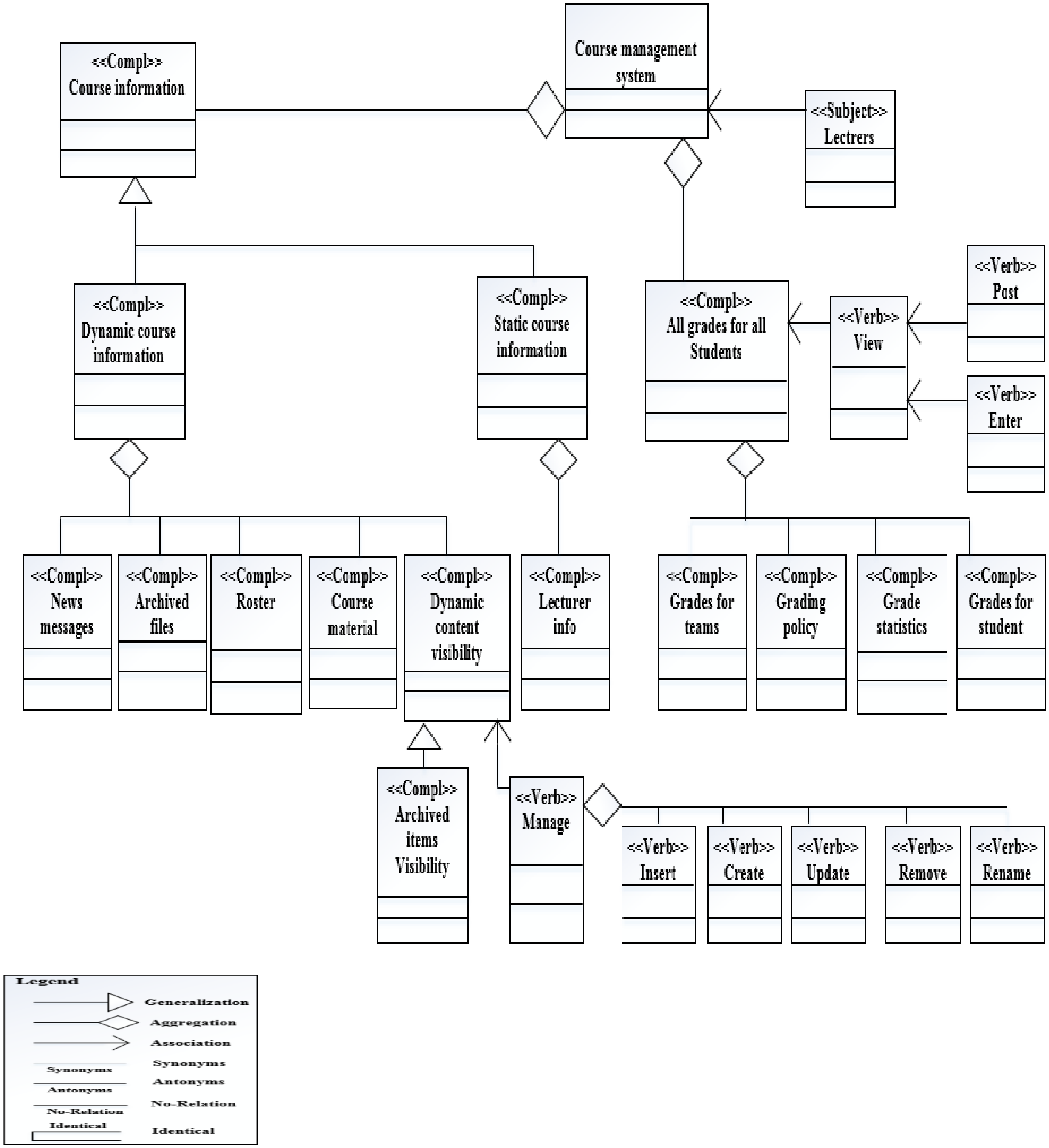
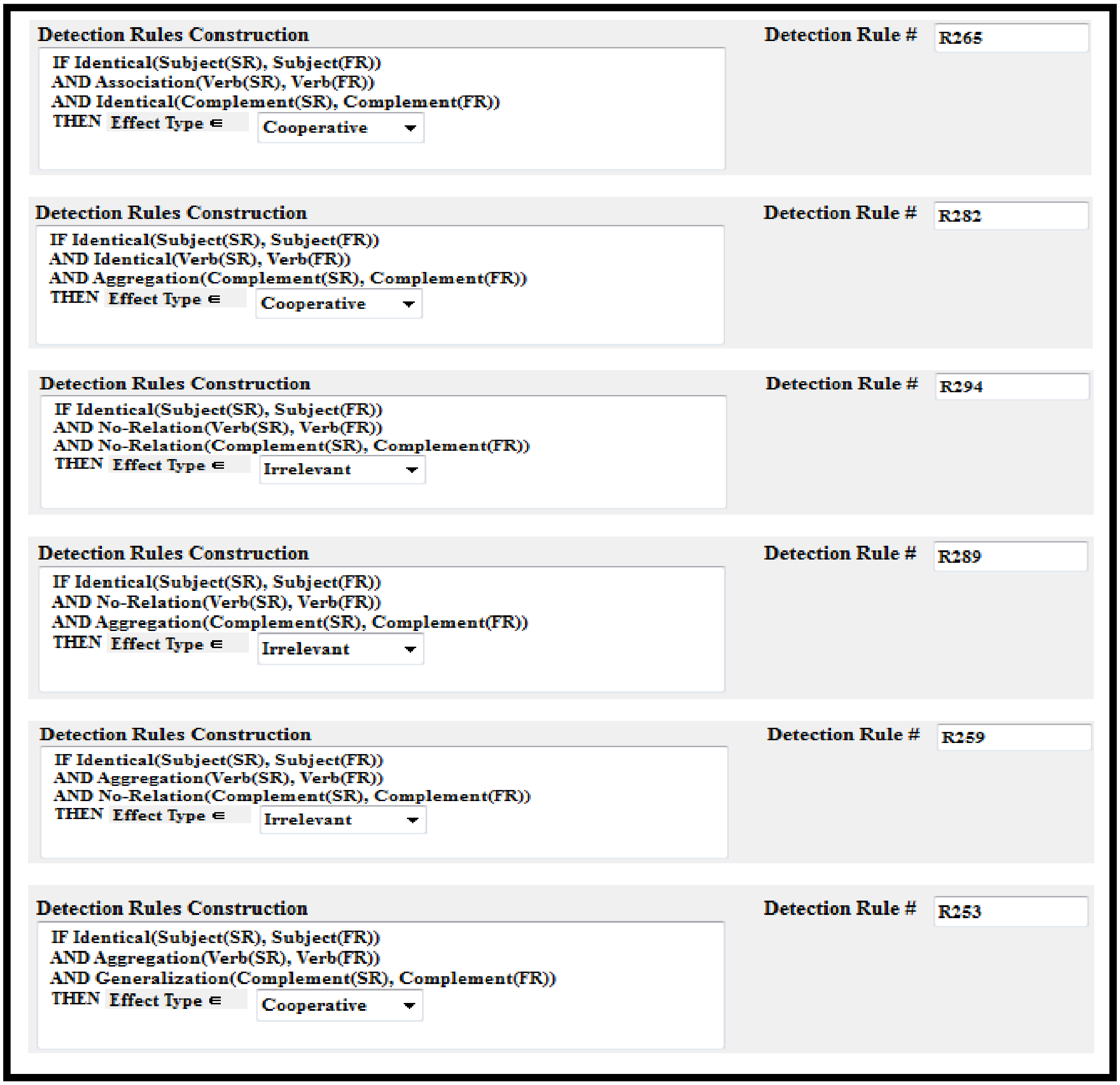


Figure 28. A part of domain ontology for lecturers system

Figure 29 shows a sample for the detection rules that have been used by our approach to detect effect types for lecturers' requirements. We show two detection rules that identify the cooperative effect: R265 and R282 while the three detection rules: R294, R289, and R259 identify the irrelevant effect.



Detection Rules Construction

Detection Rule # R265

IF Identical(Subject(SR), Subject(FR))
AND Association(Verb(SR), Verb(FR))
AND Identical(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rules Construction

Detection Rule # R282

IF Identical(Subject(SR), Subject(FR))
AND Identical(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rules Construction

Detection Rule # R294

IF Identical(Subject(SR), Subject(FR))
AND No-Relation(Verb(SR), Verb(FR))
AND No-Relation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

Detection Rules Construction

Detection Rule # R289

IF Identical(Subject(SR), Subject(FR))
AND No-Relation(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

Detection Rules Construction

Detection Rule # R259

IF Identical(Subject(SR), Subject(FR))
AND Aggregation(Verb(SR), Verb(FR))
AND No-Relation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

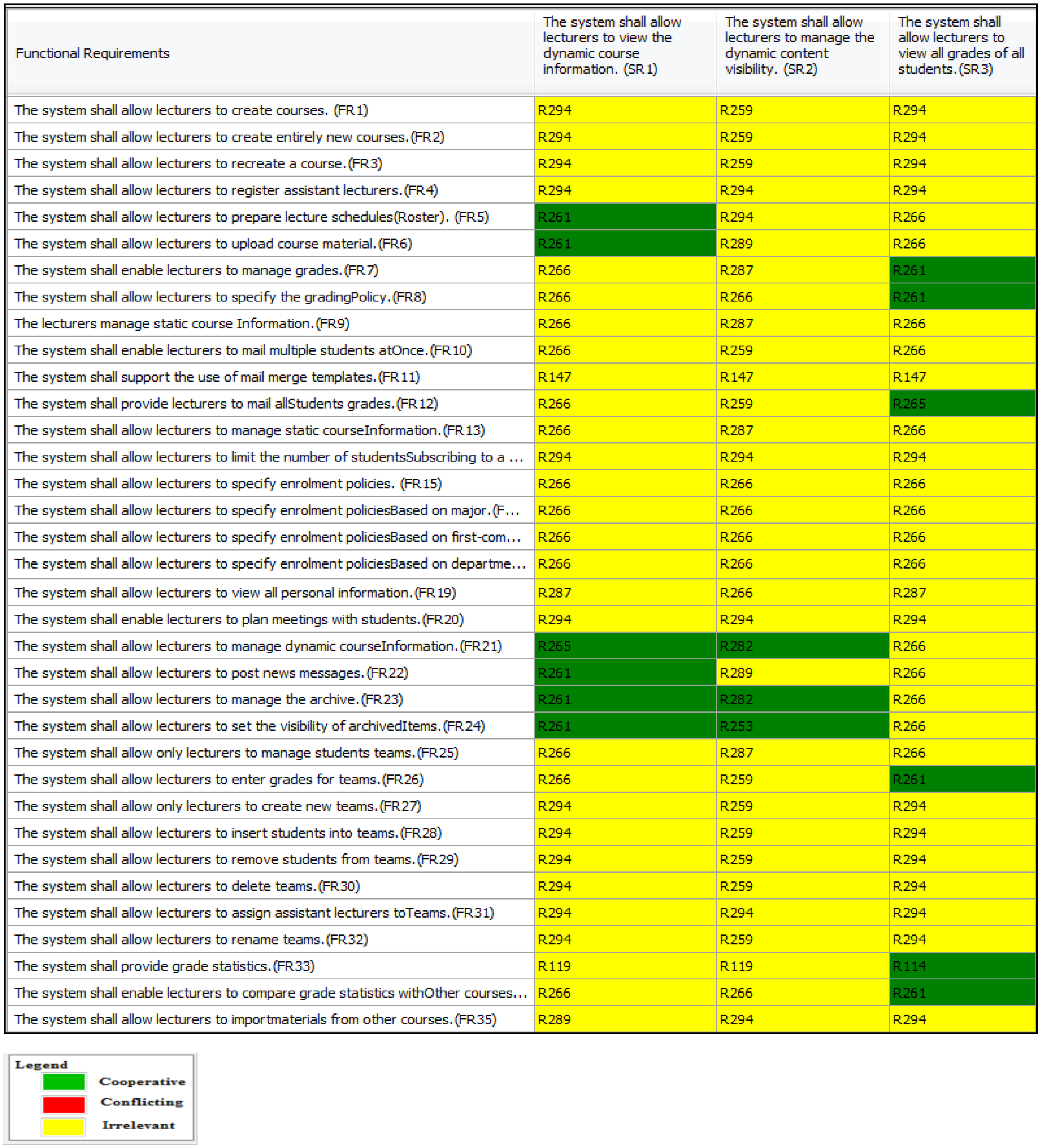
Detection Rules Construction

Detection Rule # R253

IF Identical(Subject(SR), Subject(FR))
AND Aggregation(Verb(SR), Verb(FR))
AND Generalization(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Figure 29. A sample of the detection rules for lecturers system

Since the lecturer's subsystem has three security requirements and 35 functional requirements, then it has 105 tracing links such: 15 cooperative effect, 90 irrelevant effect, and zero conflicting effects as in Figure 30.



Functional Requirements	The system shall allow lecturers to view the dynamic course information. (SR1)	The system shall allow lecturers to manage the dynamic content visibility. (SR2)	The system shall allow lecturers to view all grades of all students. (SR3)
The system shall allow lecturers to create courses. (FR1)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to create entirely new courses. (FR2)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to recreate a course. (FR3)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to register assistant lecturers. (FR4)	🟨 R294	🟨 R294	🟨 R294
The system shall allow lecturers to prepare lecture schedules(Roster). (FR5)	🟩 R261	🟨 R294	🟨 R266
The system shall allow lecturers to upload course material. (FR6)	🟩 R261	🟨 R289	🟨 R266
The system shall enable lecturers to manage grades. (FR7)	🟨 R266	🟨 R287	🟩 R261
The system shall allow lecturers to specify the gradingPolicy. (FR8)	🟨 R266	🟨 R266	🟩 R261
The lecturers manage static course Information. (FR9)	🟨 R266	🟨 R287	🟨 R266
The system shall enable lecturers to mail multiple students atOnce. (FR10)	🟨 R266	🟨 R259	🟨 R266
The system shall support the use of mail merge templates. (FR11)	🟨 R147	🟨 R147	🟨 R147
The system shall provide lecturers to mail allStudents grades. (FR12)	🟨 R266	🟨 R259	🟩 R265
The system shall allow lecturers to manage static courseInformation. (FR13)	🟨 R266	🟨 R287	🟨 R266
The system shall allow lecturers to limit the number of studentsSubscribing to a …	🟨 R294	🟨 R294	🟨 R294
The system shall allow lecturers to specify enrolment policies. (FR15)	🟨 R266	🟨 R266	🟨 R266
The system shall allow lecturers to specify enrolment policiesBased on major. (F…	🟨 R266	🟨 R266	🟨 R266
The system shall allow lecturers to specify enrolment policiesBased on first-com…	🟨 R266	🟨 R266	🟨 R266
The system shall allow lecturers to specify enrolment policiesBased on departme…	🟨 R266	🟨 R266	🟨 R266
The system shall allow lecturers to view all personal information. (FR19)	🟨 R287	🟨 R266	🟨 R287
The system shall enable lecturers to plan meetings with students. (FR20)	🟨 R294	🟨 R294	🟨 R294
The system shall allow lecturers to manage dynamic courseInformation. (FR21)	🟩 R265	🟩 R282	🟨 R266
The system shall allow lecturers to post news messages. (FR22)	🟩 R261	🟨 R289	🟨 R266
The system shall allow lecturers to manage the archive. (FR23)	🟩 R261	🟩 R282	🟨 R266
The system shall allow lecturers to set the visibility of archivedItems. (FR24)	🟩 R261	🟩 R253	🟨 R266
The system shall allow only lecturers to manage students teams. (FR25)	🟨 R266	🟨 R287	🟨 R266
The system shall allow lecturers to enter grades for teams. (FR26)	🟨 R266	🟨 R259	🟩 R261
The system shall allow only lecturers to create new teams. (FR27)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to insert students into teams. (FR28)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to remove students from teams. (FR29)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to delete teams. (FR30)	🟨 R294	🟨 R259	🟨 R294
The system shall allow lecturers to assign assistant lecturers toTeams. (FR31)	🟨 R294	🟨 R294	🟨 R294
The system shall allow lecturers to rename teams. (FR32)	🟨 R294	🟨 R259	🟨 R294
The system shall provide grade statistics. (FR33)	🟨 R119	🟨 R119	🟩 R114
The system shall enable lecturers to compare grade statistics withOther courses…	🟨 R266	🟨 R266	🟩 R261
The system shall allow lecturers to importmaterials from other courses. (FR35)	🟨 R289	🟨 R294	🟨 R294

Legend
🟩 Cooperative
🟥 Conflicting
🟨 Irrelevant

Figure 30. SFRD for lecturers system

1. Prioritization for the security requirements:

    - Weight (SR1) = 6
    - Weight (SR2) = 3
    - Weight (SR3) = 6

Since both SR1 and SR3 have the same priority, therefore, the prioritization from high to low is either (SR3, SR1, SR2) or (SR1, SR3, SR2).

2. Finding association between security requirements: We find association among security requirements as in Figure 31.

    - Association (SR1, SR2) = 3/6= 0.50
    - Association (SR1, SR3) = 0/12= 0.0
    - Association (SR2, SR3) = 0/9= 0.0

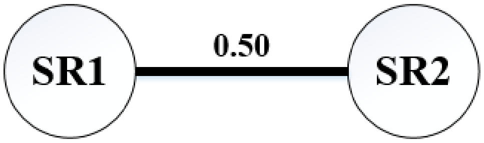


Figure 31. A weighted association graph for lecturers system

From Figure 31, we can find that SR1 associated with only SR2, and there is no an association between (SR1, SR3), (SR2, SR3). Knowing this valuable information indicates that the impact of change SR1 will affect only SR2, and it will not affect SR3. In addition, the impact of change SR2 will not affect SR3 since there is not any common parts between them.

3. Inconsistency: There is no inconsistency for this particular requirements set because it does not have any conflicting effect.

4.5. Health Monitor System

Health domain [60] is one of the most critical domain in which the security features are very critical to the health staff and to the patients, health domain is one of the complex domain that takes into account the other political and legal issues. Consequently, most of these software that developed in health domain are critical because they encompass people health in a large scale and therefore are very delicate to faults come from different requirements sources, so it is very important to trace the security and functional requirements to figure out the relationships for such systems.

In order to capture the health domain, we select to apply our approach to the health monitor system [61] which aims to help the users to check their food healthy level. Health Monitor software provides a diet chart and fitness plan to the users based on various factors that the user entered to the system which include: age, gender, race, height, weight, individual life style, food habits details, diet plans, and exercise plans.

The system help the user to follow a healthy food style and keep them away from various chronic diseases by showing several beneficial nutritional information for several types of food such as calories contents and junky food effects, It let the user to judge how healthy they are through asking several questions and a obtain facts from the user that related to the food habits, exercise, and life style.

This system requires to apply the provided web server security that related to the web services and the server database protection as well as keep the entered user data private and secured form the disclosure or other unauthorized access from hackers. This system has three security requirements and 30 functional requirements. Figure 32 shows a portion for the domain ontology for the health monitor system.

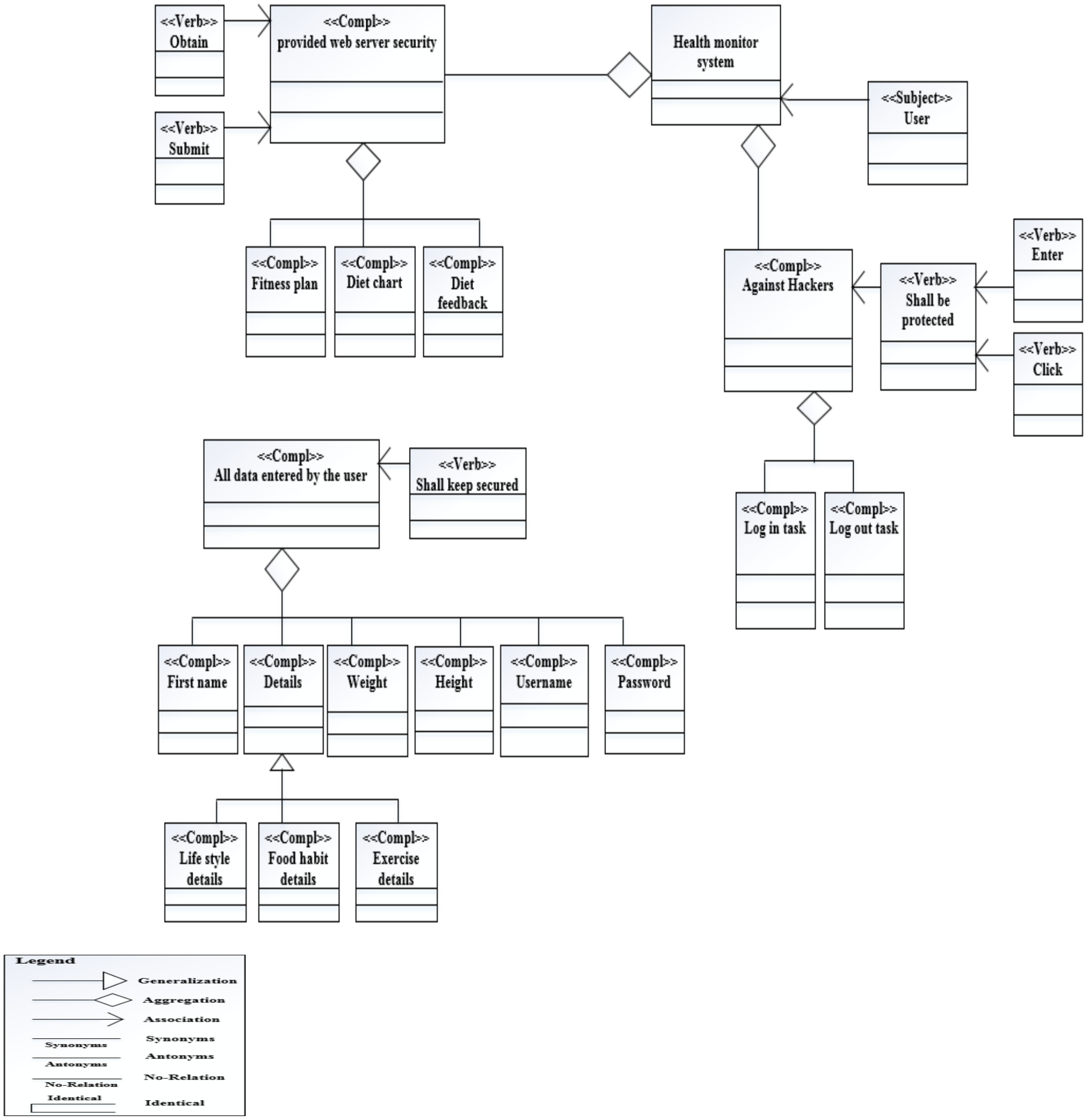
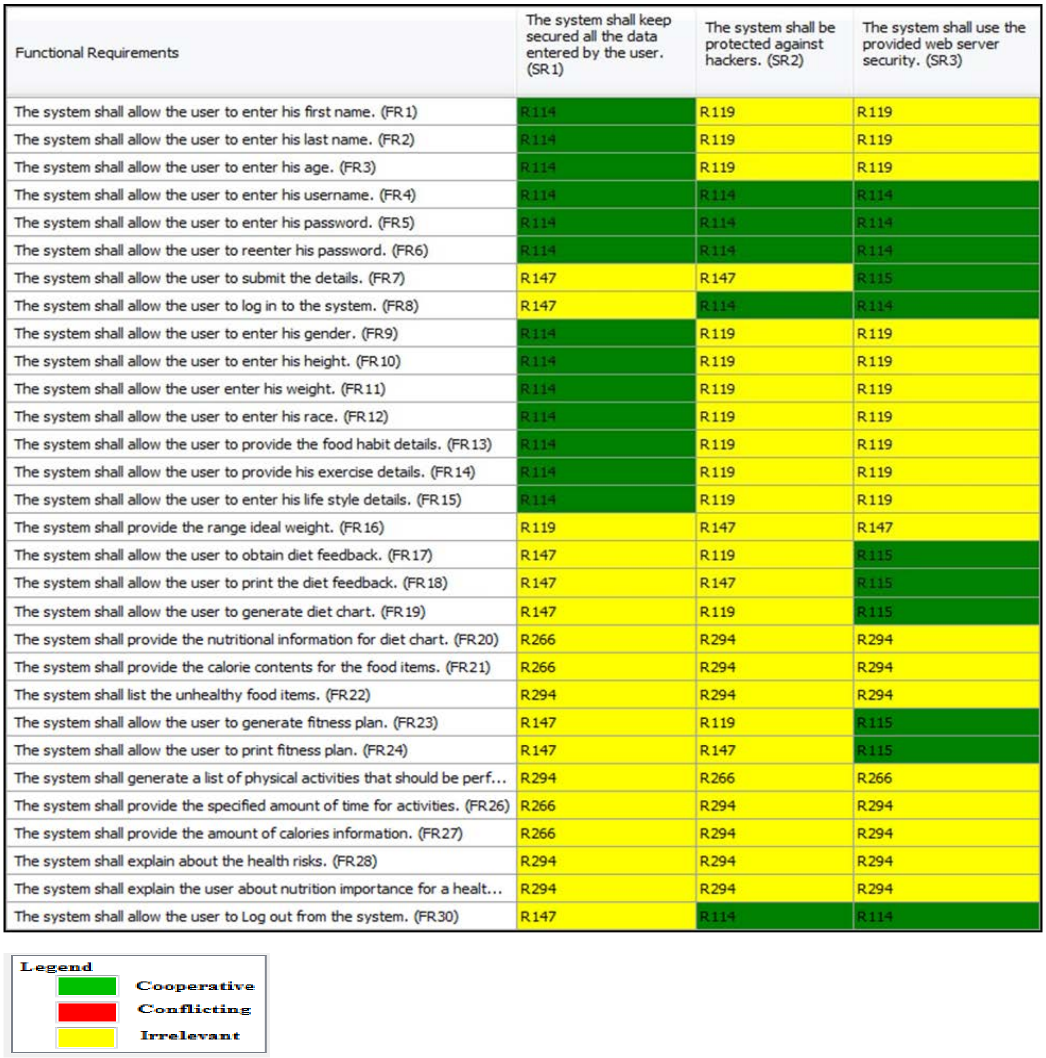


Figure 32. A part of domain ontology for health monitor system

The Figure 33 shows SFRD for lecturers has 29 cooperative effect, 31 irrelevant effect, and zero conflicting effects.



Functional Requirements	The system shall keep secured all the data entered by the user. (SR1)	The system shall be protected against hackers. (SR2)	The system shall use the provided web server security. (SR3)
The system shall allow the user to enter his first name. (FR1)	R114	R119	R119
The system shall allow the user to enter his last name. (FR2)	R114	R119	R119
The system shall allow the user to enter his age. (FR3)	R114	R119	R119
The system shall allow the user to enter his username. (FR4)	R114	R114	R114
The system shall allow the user to enter his password. (FR5)	R114	R114	R114
The system shall allow the user to reenter his password. (FR6)	R114	R114	R114
The system shall allow the user to submit the details. (FR7)	R147	R147	R115
The system shall allow the user to log in to the system. (FR8)	R147	R114	R114
The system shall allow the user to enter his gender. (FR9)	R114	R119	R119
The system shall allow the user to enter his height. (FR10)	R114	R119	R119
The system shall allow the user enter his weight. (FR11)	R114	R119	R119
The system shall allow the user to enter his race. (FR12)	R114	R119	R119
The system shall allow the user to provide the food habit details. (FR13)	R114	R119	R119
The system shall allow the user to provide his exercise details. (FR14)	R114	R119	R119
The system shall allow the user to enter his life style details. (FR15)	R114	R119	R119
The system shall provide the range ideal weight. (FR16)	R119	R147	R147
The system shall allow the user to obtain diet feedback. (FR17)	R147	R119	R115
The system shall allow the user to print the diet feedback. (FR18)	R147	R147	R115
The system shall allow the user to generate diet chart. (FR19)	R147	R119	R115
The system shall provide the nutritional information for diet chart. (FR20)	R266	R294	R294
The system shall provide the calorie contents for the food items. (FR21)	R266	R294	R294
The system shall list the unhealthy food items. (FR22)	R294	R294	R294
The system shall allow the user to generate fitness plan. (FR23)	R147	R119	R115
The system shall allow the user to print fitness plan. (FR24)	R147	R147	R115
The system shall generate a list of physical activities that should be perf...	R294	R266	R266
The system shall provide the specified amount of time for activities. (FR26)	R266	R294	R294
The system shall provide the amount of calories information. (FR27)	R266	R294	R294
The system shall explain about the health risks. (FR28)	R294	R294	R294
The system shall explain the user about nutrition importance for a healt...	R294	R294	R294
The system shall allow the user to Log out from the system. (FR30)	R147	R114	R114

Legend
🟩 Cooperative
🟥 Conflicting
🟨 Irrelevant

Figure 33. SFRD for health monitor system

1. Prioritization for the security requirements:

- Weight (SR1) = 13

- Weight (SR2) = 5

- Weight (SR3) = 11, the prioritization from high to low is: SR1, SR3, and SR2.

2. Finding association between security requirements: We find association among security requirements as in Figure 34:

    - Association (SR1, SR2) = 3/15= 0.20
    - Association (SR1, SR3) = 3/21= 0.14
    - Association (SR2, SR3) = 5/11= 0.45

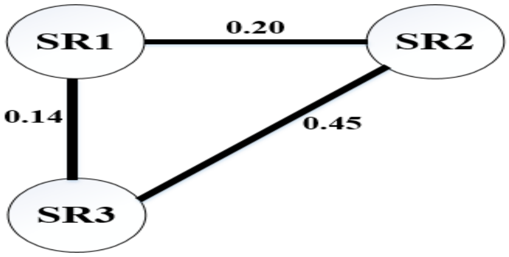


Figure 34. A weighted association graph for health monitor system

3. Inconsistency for the security requirements: there is no inconsistency for this particular requirements set because it does not have any conflicting effect.

4.6. Automated Railway Reservation System

In order to capture the booking system environment, we will apply our approach on the automated railway reservation system [62], it has been designed to provide an electronic version of the railway passenger reservation system. The system will have a user-friendly graphical interface and will be more cost effective compared to the current non-electronic version of the reservation system. This system provide customers to get their tickets in a more convenient way and control of the railway ticket sales to avoid scalping and overselling of tickets.

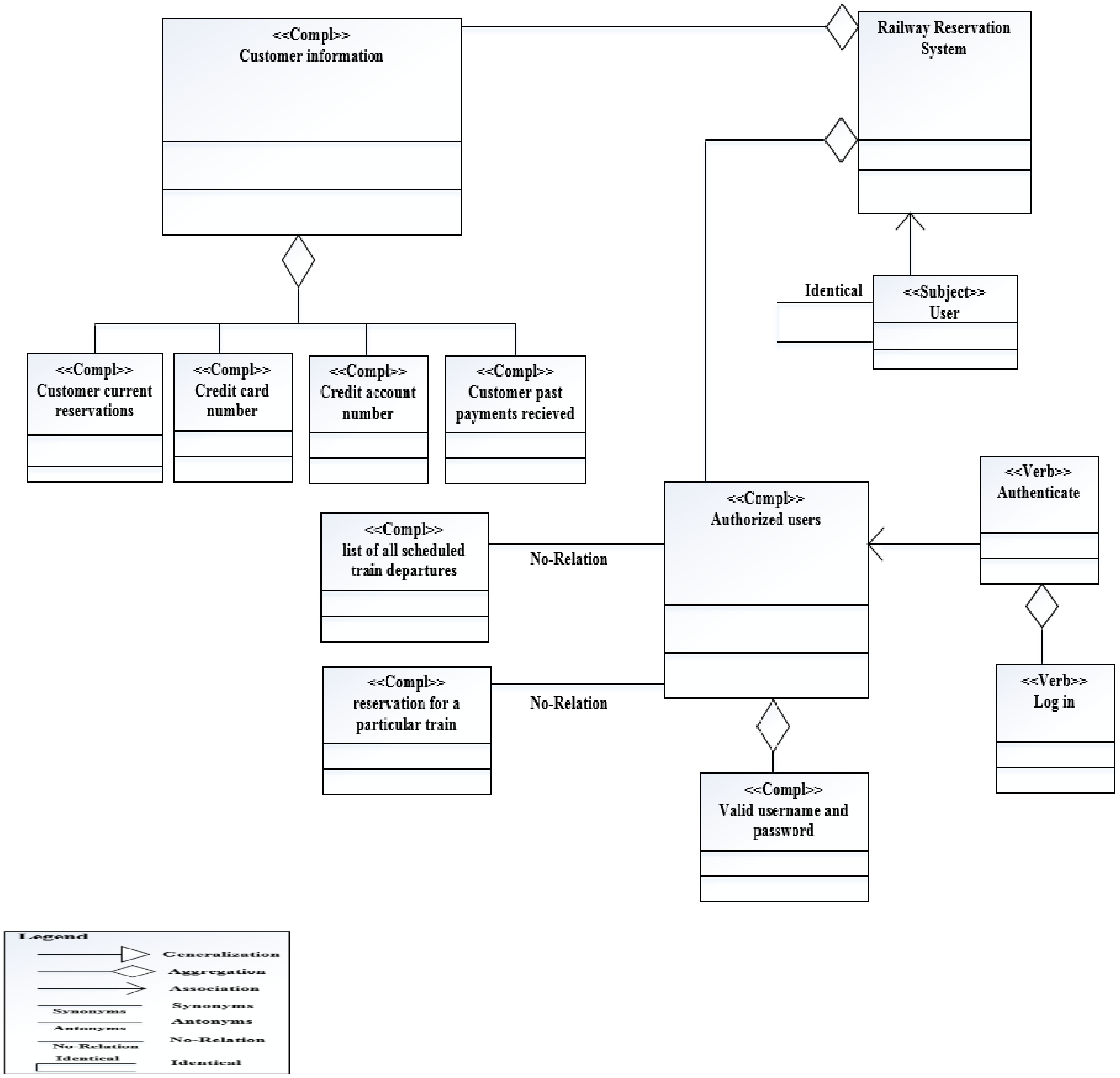
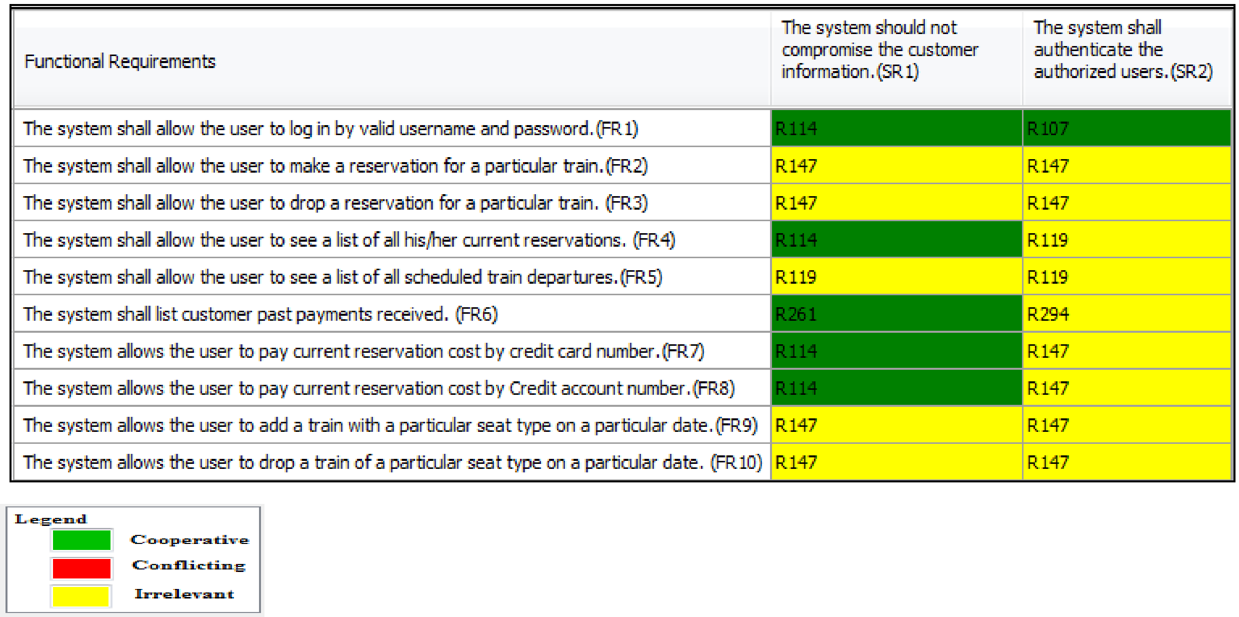


Figure 35. A part of domain ontology for railway reservation system

Figure 35 shows a portion of domain ontology for railway reservation system. This system is relatively small and has only two security requirements and 10 functional requirements. Figure 36 shows SFRD for lecturers that has 6 cooperative effect, 14 irrelevant effect, and zero conflicting effects.



Functional Requirements	The system should not compromise the customer information.(SR1)	The system shall authenticate the authorized users.(SR2)
The system shall allow the user to log in by valid username and password.(FR1)	🟩 R114	🟩 R107
The system shall allow the user to make a reservation for a particular train.(FR2)	🟨 R147	🟨 R147
The system shall allow the user to drop a reservation for a particular train. (FR3)	🟨 R147	🟨 R147
The system shall allow the user to see a list of all his/her current reservations. (FR4)	🟩 R114	🟨 R119
The system shall allow the user to see a list of all scheduled train departures.(FR5)	🟨 R119	🟨 R119
The system shall list customer past payments received. (FR6)	🟩 R261	🟨 R294
The system allows the user to pay current reservation cost by credit card number.(FR7)	🟩 R114	🟨 R147
The system allows the user to pay current reservation cost by Credit account number.(FR8)	🟩 R114	🟨 R147
The system allows the user to add a train with a particular seat type on a particular date.(FR9)	🟨 R147	🟨 R147
The system allows the user to drop a train of a particular seat type on a particular date. (FR10)	🟨 R147	🟨 R147

Legend

🟩	Cooperative
🟥	Conflicting
🟨	Irrelevant

Figure 36. SFRD for railway reservation system

1. Prioritization for security requirements:

   - Weight (SR1) = 5
   - Weight (SR2) = 1, the prioritization from high to low is: SR1, SR2.

2. Finding association between security requirements:

   - Association (SR1, SR2) = 1/5= 0.20

As a result, SR1 and SR2 positively affect the first functional requirements that describe the login functionality, it includes valid username and password (i.e., FR1) considered as a part from the customer information that should not supposed to compromise (i.e., SR1) as well as valid username and password used by the system to authenticate the authorized user (i.e., SR2).

Figure 37 shows the weighted graph for the association between SR1 and SR2, SR1 ensures to keep the user information from the disclosure, while SR2 ensures the authentication for the access to enable only the authorized users reach the system.

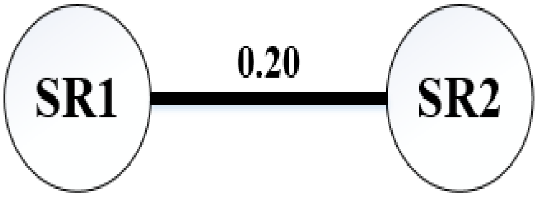


Figure 37. A weighted association graph for railway reservation system

3. Inconsistency: there is no inconsistency for this particular requirements set because it does not have any conflicting effect.

4.7. Hotel Management System

In order to capture the hospitality industry domain, we select to apply our approach to hotel management system [63] is one of the popular software solution that used in hospitality industry such as: hotels, resorts, inns, motels, and lodges. Our product hotel management system is a comprehensive software suite consisting of combined elements for various aspects of hotel management. The software product to be produced is a hotel management system which will automate the major hotel operations.

The first subsystem is a reservation which keeps track of reservations and room availability. The second subsystem is the tracking and selling food which charges the current room and meals. The third subsystem is a general management services and automated tasks which generates reports to audit all hotel operations and allows modification of subsystem.

The system will be able to handle many services to take care of all customers in a quick manner. Additionally, the system should be user appropriate, easy to use, provide easy recovery of errors and have an overall end user high subjective satisfaction.

The system has 3 security requirements and 23 functional requirements, the security requirements for hotel management system describe the access control for the hotel customer service representative and manager such as customer service representative has only access the reservation and food subsystem while the manager can access the whole three subsystems: reservation, food, and management subsystems. The manager has the privileges to add, delete, and modify the information that related to rooms, menu items, and price while the customer representative can only modify the customer reservations.

Figure 38 shows the domain ontology for the hotel management system. According to the domain ontology for hotel system, user is subject-concept and it can be customer service representative, or manager. Access is a verb-concept and it aggregates from several verb-concepts: add, delete, and modify.

Furthermore, reservation subsystem is a complement-concept and it aggregates from many complement-concepts: customer’s name, room number, number of occupants, check in date/time, and check out date/time. In the same way, food subsystem is a complement-concept and it aggregates from many complement-concepts: restaurant booking, purchasing meals, and payment for meals.

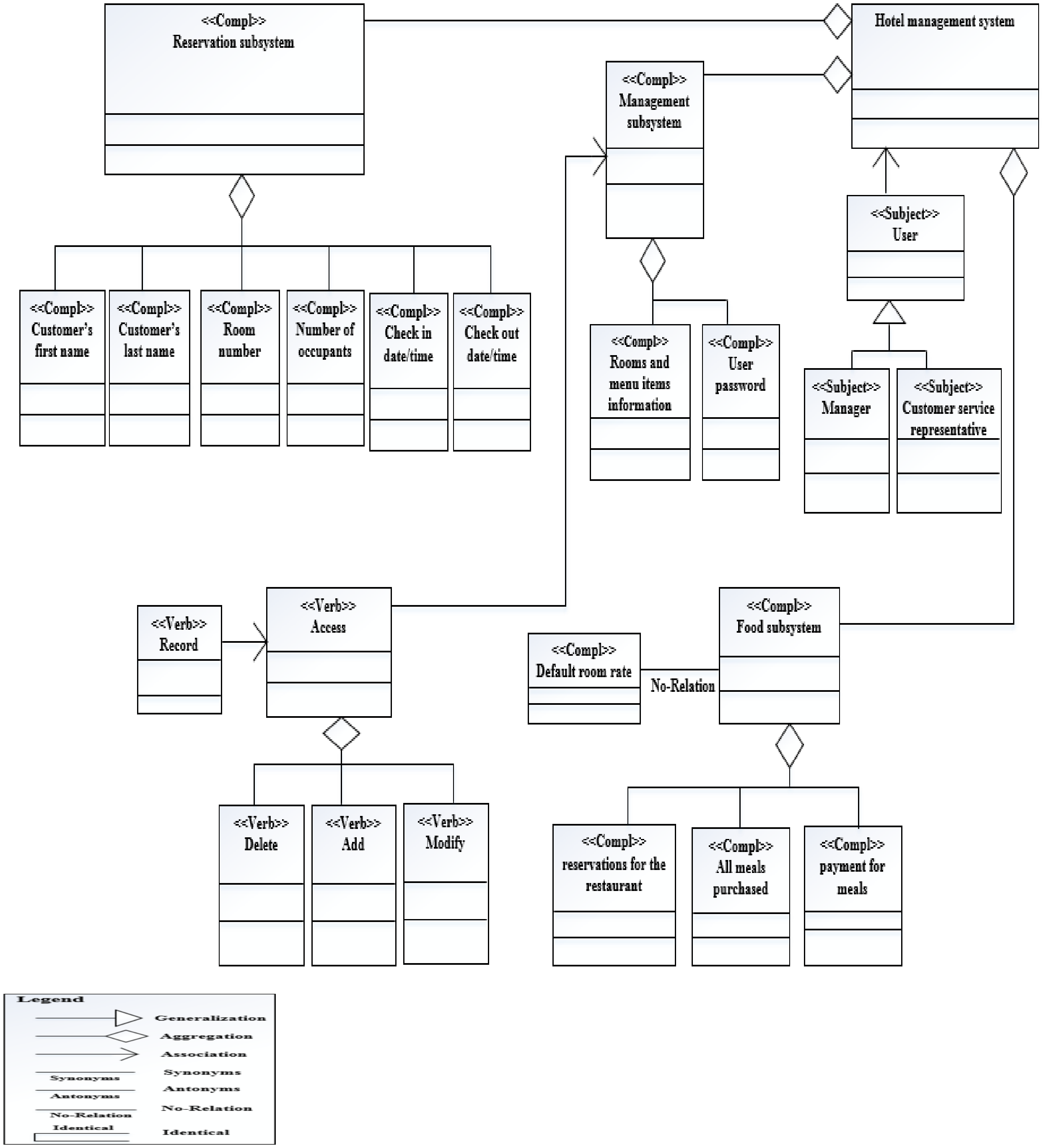
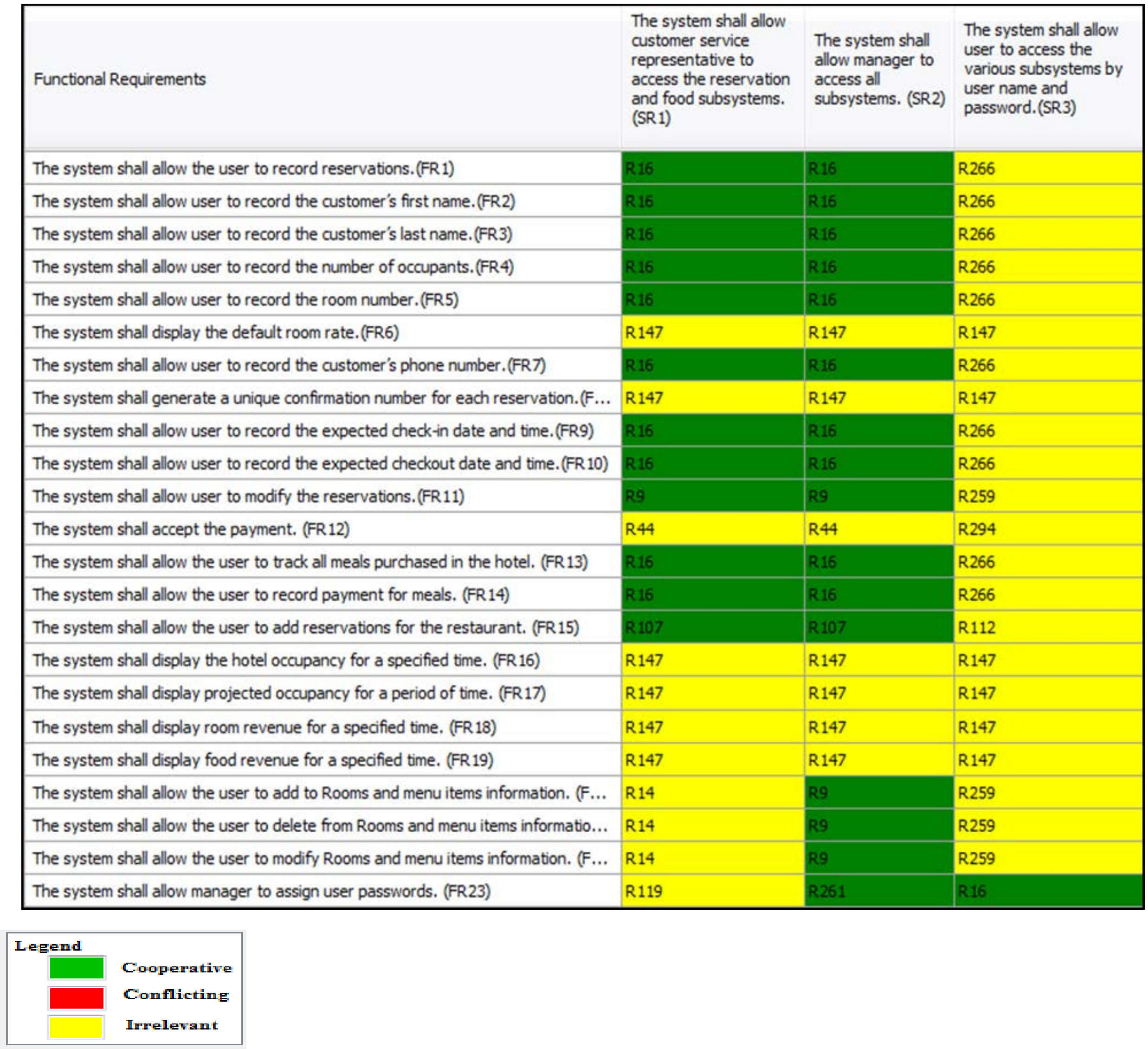


Figure 38. A part of domain ontology for hotel management system

The Figure 39 shows SFRD for hotel management system that has 29 cooperative effect, 40 irrelevant effect, and zero conflicting effect as follows:

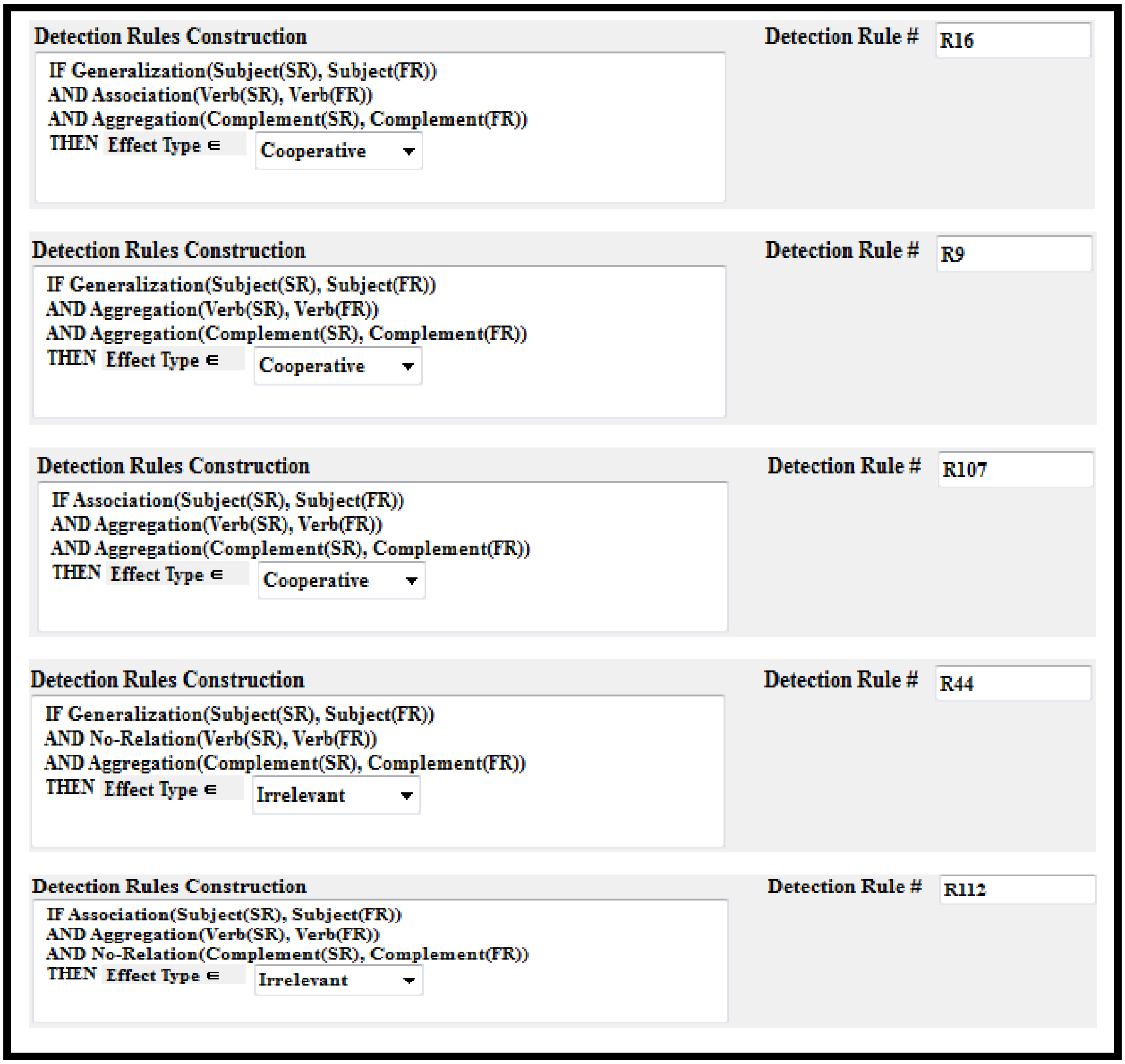


Functional Requirements	The system shall allow customer service representative to access the reservation and food subsystems. (SR1)	The system shall allow manager to access all subsystems. (SR2)	The system shall allow user to access the various subsystems by user name and password. (SR3)
The system shall allow the user to record reservations. (FR1)	🟩 R16	🟩 R16	🟨 R266
The system shall allow user to record the customer's first name. (FR2)	🟩 R16	🟩 R16	🟨 R266
The system shall allow user to record the customer's last name. (FR3)	🟩 R16	🟩 R16	🟨 R266
The system shall allow user to record the number of occupants. (FR4)	🟩 R16	🟩 R16	🟨 R266
The system shall allow user to record the room number. (FR5)	🟩 R16	🟩 R16	🟨 R266
The system shall display the default room rate. (FR6)	🟨 R147	🟨 R147	🟨 R147
The system shall allow user to record the customer's phone number. (FR7)	🟩 R16	🟩 R16	🟨 R266
The system shall generate a unique confirmation number for each reservation. (F...	🟨 R147	🟨 R147	🟨 R147
The system shall allow user to record the expected check-in date and time. (FR9)	🟩 R16	🟩 R16	🟨 R266
The system shall allow user to record the expected checkout date and time. (FR10)	🟩 R16	🟩 R16	🟨 R266
The system shall allow user to modify the reservations. (FR11)	🟩 R9	🟩 R9	🟨 R259
The system shall accept the payment. (FR12)	🟨 R44	🟨 R44	🟨 R294
The system shall allow the user to track all meals purchased in the hotel. (FR13)	🟩 R16	🟩 R16	🟨 R266
The system shall allow the user to record payment for meals. (FR14)	🟩 R16	🟩 R16	🟨 R266
The system shall allow the user to add reservations for the restaurant. (FR15)	🟩 R107	🟩 R107	🟨 R112
The system shall display the hotel occupancy for a specified time. (FR16)	🟨 R147	🟨 R147	🟨 R147
The system shall display projected occupancy for a period of time. (FR17)	🟨 R147	🟨 R147	🟨 R147
The system shall display room revenue for a specified time. (FR18)	🟨 R147	🟨 R147	🟨 R147
The system shall display food revenue for a specified time. (FR19)	🟨 R147	🟨 R147	🟨 R147
The system shall allow the user to add to Rooms and menu items information. (F...	🟨 R14	🟩 R9	🟨 R259
The system shall allow the user to delete from Rooms and menu items informatio...	🟨 R14	🟩 R9	🟨 R259
The system shall allow the user to modify Rooms and menu items information. (F...	🟨 R14	🟩 R9	🟨 R259
The system shall allow manager to assign user passwords. (FR23)	🟨 R119	🟩 R261	🟩 R16

Legend
🟩 Cooperative
🟥 Conflicting
🟨 Irrelevant

Figure 39. SFRD for hotel management system

As we can notice from the above SFRD, SR1 and SR2 affected several common mutual functional requirements because SR1 consider the access to reservation and food subsystems while SR2 includes SR1 plus the access to management subsystem. In this particular case, this means that both requirements are strongly associated, while SR1 and SR3 are weakly associated. Figure 40 shows a sample for the used detection rules that have been used to capture the relationship between security and functional requirements.



Detection Rules Construction

IF Generalization(Subject(SR), Subject(FR))
AND Association(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rule # R16

Detection Rules Construction

IF Generalization(Subject(SR), Subject(FR))
AND Aggregation(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rule # R9

Detection Rules Construction

IF Association(Subject(SR), Subject(FR))
AND Aggregation(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Cooperative

Detection Rule # R107

Detection Rules Construction

IF Generalization(Subject(SR), Subject(FR))
AND No-Relation(Verb(SR), Verb(FR))
AND Aggregation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

Detection Rule # R44

Detection Rules Construction

IF Association(Subject(SR), Subject(FR))
AND Aggregation(Verb(SR), Verb(FR))
AND No-Relation(Complement(SR), Complement(FR))
THEN Effect Type ∈ Irrelevant

Detection Rule # R112

Figure 40. A sample of the detection rules for hotel management system

1. Prioritization for the security requirements:

- Weight (SR1) = 12

- Weight (SR2) = 16

- Weight (SR3) = 1, the prioritization from high to low is: SR2, SR1, and SR3.

2. Finding association between security requirements: We find association among security requirements as in Figure 41:
    - Association (SR1, SR2) = 12/16 = 0.75
    - Association (SR1, SR3) = 0/13= 0.0
    - Association (SR2, SR3) = 1/17= 0.05

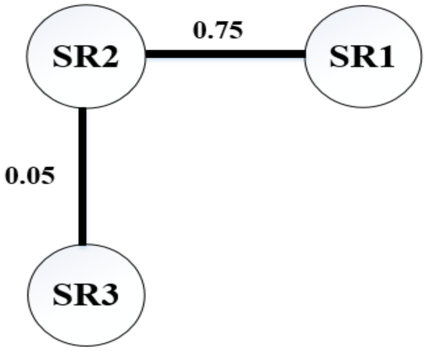


Figure 41. A weighted association graph for hotel management system

As we can see in Figure 41, since SR1 considers reservation and food subsystems while SR2 considers reservation, food, and management subsystems, then both SR1 and SR2 are highly dependent on each other such as SR2 contains SR1.

Our approach can be applied to various types of requirements domain, in this research we show how we can apply our traceability approach for four different domains: health, educational, commercial, and hospitality industry. Table 7 statistically summarizes the number for security requirements, functional requirements, number of cooperative, number of conflicting, and number of irrelevant effects, and number of generated tracing links correspondingly for each system.

Table 7. Statistical summary for all systems

System	No. of SR	No. of FR	No. of Cooperative	No. of Conflicting	No. of Irrelevant	No. of Tracing links
Online Medical	4	33	19	4	109	132
Online Store	3	63	16	2	171	189
Students	4	17	12	0	56	68
Lecturers	3	35	15	0	90	105
Health monitor	3	30	29	0	61	90
Railway reservation	2	10	6	0	14	20
Hotel management	3	23	29	0	40	69

We find that online medical database system and health monitor system are the most critical systems since they belong to the health domain. Among all seven systems, we can find that both of health monitor and hotel management system have the largest number for cooperative effect, online medical system has the largest number for conflicting effect. Moreover, online store system has the largest number for irrelevant effect.

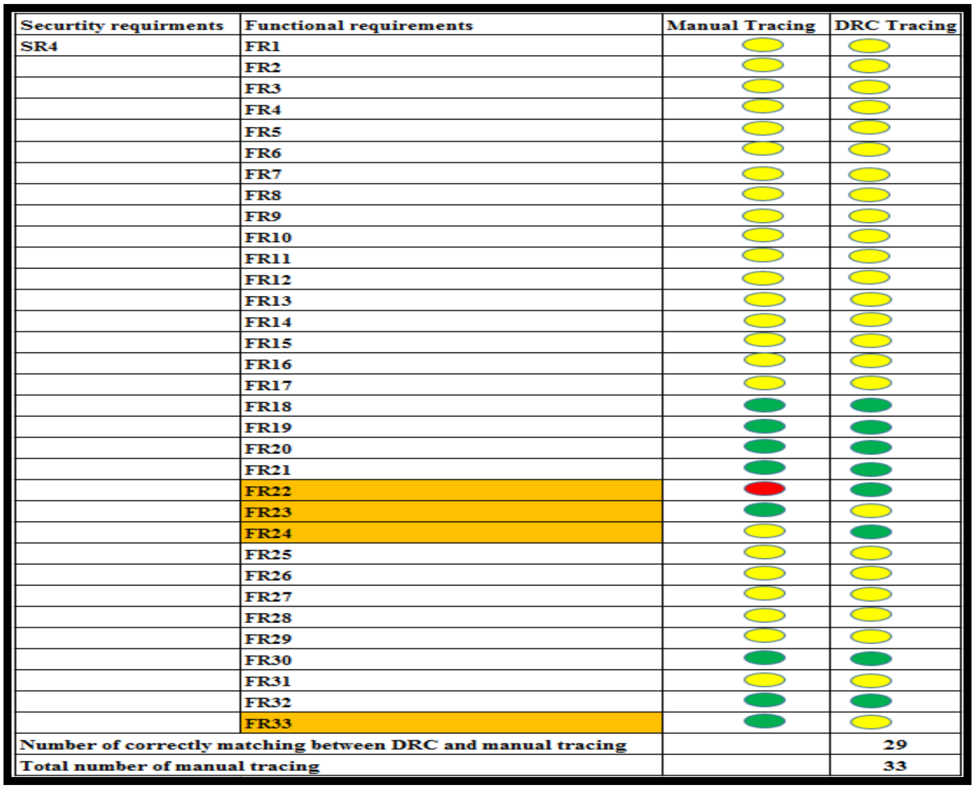
4.8. Evaluation for DRC Tool

Several metrics have been used to assess the quality of the automated tracing tools by comparing with the manual tracing. The manual tracing [64] is still widely used approach in requirements traceability, and it considers the human judgment based on the domain knowledge. Today, manual traceability method is still preferred by a significant percentage of software industry because in requirements engineering phase.

We cannot avoid or ignore the knowledge for the experts who have several years of experience in this particular domain. To evaluate the results of our tool with the manual tracing results, we decide to use accuracy [65] which measures the degree to which a requirements tracing tool returns all the correct matching effect types with the manual tracing as in Equation 3:

$$\text{Accuracy} = \frac{\text{Number of correctly matching effects from manual and tool tracing}}{\text{Total number of effects from manual tracing}} \qquad \text{(Eq. 3)}$$

We evaluate each of security requirement for all systems. For example, Figure 42 shows the comparison results between manual and tool tracing of SR4 for an online medical database system.



Securtity requirments	Functional requirements	Manual Tracing	DRC Tracing
SR4	FR1	🟡	🟡
	FR2	🟡	🟡
	FR3	🟡	🟡
	FR4	🟡	🟡
	FR5	🟡	🟡
	FR6	🟡	🟡
	FR7	🟡	🟡
	FR8	🟡	🟡
	FR9	🟡	🟡
	FR10	🟡	🟡
	FR11	🟡	🟡
	FR12	🟡	🟡
	FR13	🟡	🟡
	FR14	🟡	🟡
	FR15	🟡	🟡
	FR16	🟡	🟡
	FR17	🟡	🟡
	FR18	🟢	🟢
	FR19	🟢	🟢
	FR20	🟢	🟢
	FR21	🟢	🟢
	FR22	🔴	🟢
	FR23	🟢	🟡
	FR24	🟡	🟢
	FR25	🟡	🟡
	FR26	🟡	🟡
	FR27	🟡	🟡
	FR28	🟡	🟡
	FR29	🟡	🟡
	FR30	🟢	🟢
	FR31	🟡	🟡
	FR32	🟢	🟢
	FR33	🟢	🟡
Number of correctly matching between DRC and manual tracing			29
Total number of manual tracing			33

Figure 42. Manual and DRC tracing of SR4 for an online medical database system

We can see that there are 29 matching, and 4 non-matching nodes. As a result, according to Formula 3, the accuracy for SR4 is 0.87. The requirements that highlighted in orange represent the non-matching nodes between DRC and manual tracing: FR22, FR23, FR24, and FR33. To show the accuracy of all security requirements for all systems. We find the accuracy for each single security requirement by comparing the matching between the results of manual tracing and tool tracing. Table 8, Table 9, Table 10, Table 11, Table 12, Table 13, and Table 14 show the accuracy ratio for security requirements for an online medical database, online store, students, lecturers, health monitor, railway reservation, and hotel management systems respectively.

Table 8. Accuracy for online medical database security requirements

Security requirements	**Matching between DRC and manual tracing**	**Accuracy**
SR1	27	27/*33= 0.81
SR2	26	26/33= 0.78
SR3	25	25/33= 0.75
SR4	24	29/33= 0.87

*No. of effects for manual tracing = 33

Table 9. Accuracy for online store security requirements

Security requirements	**Matching between DRC and manual tracing**	**Accuracy**
SR1	30	30/*35= 0.85
SR2	30	30/35= 0.85
SR3	29	29/35= 0.82

*No. of effects for manual tracing = 35

Table 10. Accuracy for student’s security requirements

Security requirements	Matching between DRC and manual tracing	Accuracy
SR1	13	13/*17= 0.76
SR2	16	16/17= 0.94
SR3	14	14/17= 0.82
SR4	13	13/17= 0.76

*No. of effects for manual tracing = 17

Table 11. Accuracy for lecturer’s security requirements

Security requirements	Matching between DRC and manual tracing	Accuracy
SR1	26	26/*36= 0.72
SR2	27	27/36= 0.75
SR3	31	31/36= 0.86

*No. of effects for manual tracing = 36

Table 12. Accuracy for health monitor security requirements

Security requirements	Matching between DRC and manual tracing	Accuracy
SR1	26	26/*30= 0.86
SR2	23	23/30= 0.76
SR3	25	25/30= 0.83

*No. of effects for manual tracing = 30

Table 13. Accuracy for railway reservation security requirements

Security requirements	Matching between DRC and manual tracing	Accuracy
SR1	9	9/*10= 0.90
SR2	9	9/10= 0.90

*No. of effects for manual tracing = 10

Table 14. Accuracy for hotel management security requirements

Security requirements	Matching between DRC and manual tracing	Accuracy
SR1	19	19/*23= 0.82
SR2	18	18/23= 0.78
SR3	17	17/23= 0.73

*No. of effects for manual tracing = 23

The experiments with our tool has indicated that it is capable of generating effect types at reasonable accuracy rates, In particular with respect to accuracy, we have identified the need to propose new effect types to capture the effect types that the existing detection rules fail to identify.

Based on the results that we obtain from the above tables, we can summarize that by using our tool, we can get an overall accuracy 80% comparing to the manual tracing which reflects a very acceptable accuracy level according to the study in [66]. We observe like different systems have different accuracy values due to several reasons such as: (1) the human errors that occurs in the manual tracing, (2) coverage for the detection rules that used in our tool, and (3) domain nature for the requirements.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

In this research, we introduce a new hybrid traceability approach for using syntactic parsing, domain ontology, and a rule based system. Our 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.

We introduced DRC 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. DRC automatically captures the effect of security requirements upon functional requirements based on using predefined detection rules. Our approach allows the users to construct several detection rules to discover relationships between security and functional requirements by combining the syntactic and semantic analysis of requirements.

We presented a several case studies of using DRC, which showed how using DRC led to improve inconsistency checking and better understanding of the inter requirements traceability. A valuable feature of our approach is that users can create domain ontology, and identify effect types based on the user-defined detection rules. DRC shows several benefits comparing with the current inter-requirements traceability tools by relating functional and non-functional requirements, captures syntactic and semantic aspects of requirements.

Also, it considers consistency checking of the relations, and provide a several reasoning rules for each identified relationships between security and functional requirements. By using our DRC tool, we get an 80% accuracy comparing with the manual tracing.

5.2. Future Work

Our future work includes to extend DRC to cover the other non-functional requirements effects onto functional ones. As a result, we can apply it for large applications. We plan to show the effects for the maintainability, availability, portability, and recoverability requirements upon functional requirements by proposing several additional detection rules, as well as, extend our domain ontology by adding more semantic concepts and relations that will capture various types of requirements domains.

In addition, we plan to propose a specific detection rules that control the impact of requirements change. We will focus on traceability impact analysis which concentrates on traceability links as the main key to express relationships between security and functional requirements. The analysis of the impact of security requirement changes onto other functional requirements can be based on requirements traceability. Requirements effect types (i.e., dependencies types) can be used as trace links to identify the impact of requirements change. The goal of impact analysis in our research is to identify which security or functional are either explicitly or implicitly affected by this particular proposed change. We intend to consider two different types of changes, the first proposed type is changes in security and functional requirements: (1) adding new requirement, (2) deleting current requirement, and (3) modifying current requirement. The second proposed type is changes in effect types: (1) adding new effect type, (2) deleting current effect type, and (3) modifying current effect type.

REFERENCES

[1] I. Sommerville, “Software Engineering,” 7th Ed. Pearson Addison Wesley, 2004.

[2] P. Zave, “Classification of research efforts in requirements engineering,” ACM Computing Surveys, vol.29, no.4, pp. 315-321, 1997.

[3] C. Hoare, “The emperor’s old clothes,” Communications of the ACM, vol.24, no.2, pp. 75-83, 1981.

[4] F. Brooks, “No Silver Bullet: Essence and Accident in Software Engineering”, IEEE Computer, vol.20, no.4, pp. 10-20, 1987.

[5] K. Emam and N. Madhavji, “Elements of Software Process Assessment and Improvement,” 1st Ed. Wiley-IEEE Computer Society Press, 1999.

[6] L. Jiang, “A framework for Requirements Engineering Process Development,” Ph.D. thesis, University of Calgary, Canada, 2005.

[7] O. Gotel and A. Finkelstein, “An Analysis of the Requirements Traceability Problem,” First IEEE International Conference on Requirements Engineering, Colorado Springs, pp. 94-101, 1994.

[8] G. Kotonya and I. Sommerville, “Requirements Engineering: Processes and Techniques,” John Wiley & Sons, 1998.

[9] K. Pohl, “Requirements Engineering: Fundamentals, Principles, and Techniques,” 1st Ed. Springer Publishing Company, 2010.

[10] W. Robinson, S. Pawlowski, and V. Volkov, “Requirements Interaction Management,” ACM Computing Surveys, vol.35,no.2, pp.132-190, 1999.

[11] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt, “An Industrial Survey of Requirements Interdependencies in Software Product Release Planning,” 5th IEEE International Symposium, pp. 84-91, 2001.

[12] J. Karlsson, S. Olsson, and K. Ryan, “Improved Practical Support for Large-scale Requirements Prioritization,” Requirements Engineering Journal, vol.2, no.1, pp.51-60, 1997.

[13] B. Ramesh, and M. Jarke, “Toward Reference Models for Requirements Traceability,” IEEE Transactions on Software Engineering, vol.27, no.1, pp.58-93, 2001.

[14] J. Karlsson and K. Ryan, “Prioritizing Requirements Using a Cost-Value Approach,” IEEE Software, vol.14,no.5,pp.67-74, 1997.

[15] K. Pohl, “Process Centered Requirements Engineering,” John Wiley & Sons, 1996.

[16] A. Dahlstedt and A. Persson, “Engineering and Managing Software Requirements,” 1st Ed. Springer, 2005.

[17] P. Carlshamre, “Release Planning in Market-Driven Software Product Development: Provoking an Understanding,” Requirements Engineering, vol.7, no.3, pp.139-151, 2002.

[18] A. Goknil, I. Kurtev, and K. Berg, “Change Impact Analysis Based on Formalization of Trace Relations for Requirements,” ECMDA Traceability Workshop, pp.59-75, 2008.

[19] F. Pinheiro, “Requirements Traceability,” Requirements Traceability in Perspectives on Software Requirements, Kluwer Academic Publishers, pp.91-113, 2004.

[20] A. Lamsweerde, “Requirements Engineering in the Year 00: A Research. Perspective,” 22nd International Conference on Software Engineering, pp.5-19, 2000.

[21] H. Kaiya and M. Saeki, “Using Domain Ontology as Domain Knowledge for Requirements Elicitation,” IEEE International Requirements Engineering Conference, pp.186–195, 2006.

[22] L. Zong, W. Zhi, Y. Ying, W. Yue, and L. Ying, “Towards a multiple ontology framework for requirements elicitation and reuse,” 31st Annual International Computer Software and Applications Conference, pp.189-195, 2007.

[23] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej, “Deriving Traceability Relationships of Multiperspective Software Artifacts from Ontology Matching,” 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel Computing, IEEE, pp.549-554, 2009.

[24] M. Jyothilakshmi and P. Samuel, “Domain Ontology Based Class Diagram Generation from Functional Requirements,” International Journal of Computer Information Systems and Industrial Management Applications, pp. 380-385, 2012.

[25] S. Lee, D. Muthurajan, R. Gandhi, et al., “Building Decision Support Problem Domain Ontology from Natural Language Requirements for Software Assurance,” International Journal of Software Engineering and Knowledge Engineering, pp. 851-884, 2006.

[26] C. Lopez, L. Cysneiros, and H. Astudillo. “NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge,” 1st International Workshop on Managing Requirements Knowledge, pp.1–10, 2008.

[27] R. Falbo, G. Guizzardi, K. Duarte, “An Ontological Approach to Domain Engineering,” 14th International Conference on Software Engineering and Knowledge Engineering, pp. 351-358, 2002.

[28] I. Jureta, J. Mylopoulos, and S. Faulkner, “Revisiting the Core Ontology and Problem in Requirements Engineering,” 16th IEEE International Conference Requirements Engineering, pp. 71-80, 2008.

[29] M. Sasikumar, S. Ramani, S. Raman, et al., “A Practical Introduction to Rule Based Expert Systems,” Narosa Publishing House, 2007.

[30] 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, 2004.

[31] X. Liu, “Fuzzy Requirements,” IEEE Potentials,vol.17, no.2, pp. 24–26, 1998.

[32] C. Temponi, J. Yen, and W. Tiao, A. “House of Quality: A fuzzy Logic-Based Requirements Analysis,” European Journal of Operational Research, vol.117, no.2, pp.340-354, 1999.

[33] J. Lee and N. Xue, “Analyzing User Requirements by Use Cases: a Goal-Driven Approach,” IEEE Software, vol.16, no.4, pp. 92-101, 1999.

[34] C. Haley, R. Laney, J. Moffett, et al., “Arguing Satisfaction of Security Requirements,” Idea Group Publishing, 2007.

[35] G. Kotonya, and I. Sommerville, “Requirements Engineering: Processes and Techniques,” John Wiley and Sons, 1998.

[36] J. Rushby, "Security Requirements Specifications: How and What?" Symposium on Requirements Engineering for Information Security, Vol. 441, 2001.

[37] T. Adams, N. Koncz, and A. Vonderohe, 2000, “Functional Requirements for a Comprehensive Transportation Location Referencing System,” North American Travel Monitoring Exhibition and Conference, 2000.

[38] R. Malan and D. Bredemeyer, “Functional Requirements and Use Cases,” Architecture Resources for Enterprise Advantage, 2001.

[39] D. Firesmith, “Engineering Security Requirements,” Journal of Object Technology, vol. 2, no. 1,pp. 53-68, 2003.

[40] IBM Rational RequisitePro. http://www-01.ibm.com/software/awdtools/reqpro/

[41] IBM Telelogic Doors. http://www.telelogic.com/Products/doors/doors/index.cfm

[42] OMG: SysML Specification. OMG ptc/06-05-04, http://www.sysml.org/specs.htm

[43] TopTeamAnalyst.http://www.technosolutions.com/topteam_requirements_management.htm

[44] B. Boehm and H. In, “Identifying Quality-Requirements Conflicts,” IEEE Software, pp. 25-35, 1996.

[45] http://www.link.cs.cmu.edu/link/submit-sentence-4.html

[46] http://wordnetweb.princeton.edu/perl/webwn

[47] http://cs.joensuu.fi/pages/koles/oop/OOP_Development.ppt

[48] http://www.englishclub.com/vocabulary/synonyms-Antonyms.htm

[49] H. Kaiya and M. Saeki, “Using Domain Ontology as Domain Knowledge for Requirements Elicitation,” 14th IEEE International Conference Requirements Engineering, pp. 186–195, 2006.

[50] R Torkar, T. Gorschek, R. Feldt, et al., “Requirements Traceability State of the Art: A Systematic Review and Industry Case Study,” Information and Software Technology Journal, 2009.

[51] H. Hofmann and F. Lehner, “Requirements Engineering as a Success Factor in Software Projects,” IEEE Software, vol.18, no.4, pp. 58-66, 2001.

[52] I. Hooks, and K. Farry, “Customer-Centered Products: Creating Successful Products through Smart Requirements Management,” American Management Association, 2001.

[53] V. Tran, B. Hummel, D. Liu, et al., “Understanding and Managing the Relationship Between Requirement Changes and Product Constraints in Component-based Software Projects,” 31st Hawaii International Conference On System Sciences, pp. 132-142, 1998.

[54] S. Lauesen, and O. Vitner, “Preventing Requirement Defects,” Requirements Engineering Journal ,vol.6,no.1, pp. 37-50, 2001.

[55] S. Nivattanakul, J. Singthongchai, E. Naenudorn, et al., “Using of Jaccard Coefficient for Keywords Similarity,” International Multi Conference of Engineers and Computer Scientists, pp. 380-384, 2013.

[56] H. Kaiya and M. Saeki, “Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach,” 5th International Conference on Quality Software, pp.223-230, 2005.

[57] J. Brackett, “Software Requirements,” Technical Report, Software Engineering Institute, 1990.

[58] http://www.utdallas.edu/~chung/.../SRS4.0.doc

[59] A. Goknil, I. Kurtev, K. van den Berg, et al., “Semantics of Trace Relations in Requirements Models for Consistency Checking and Inferencing,” Software and Systems Modeling, vol.10, no.1, pp. 31-54, 2011.

[60] L. Cysneiros, “Requirements Engineering in the Health Care Domain,” International Conference on Requirements Engineering, pp.350-356, 2002.

[61] http://www.cmaps.cmappers.net/rid=1HZ2S6K7D...GTX/SRS_HealthMonitor.doc

[62] http://www.oocities.org/cs5391/SRS4.htm

[63] http://www.oocities.org/swe626/HotelManagementSystemCorrectFinalSRS.doc

[64] K. Andrew and H. Saiedian. “Why Software Requirements Traceability Remains a Challenge,” The Journal of Defense Software Engineering, 2009.

[65] J. Hayes, A. Dekhtyar, and S. Sundaram, “Advancing Candidate Link Generation for Requirements Tracing: The study of Methods,” IEEE Transactions on Software Engineering, vol.32, no.1, pp. 4-19, 2006.

[66] D. Cuddeback, A. Dekhtyar, and J. Hayes, “Automated requirements traceability: The Study of Human Analysts,” International Requirements Engineering Conference, pp.231-240, 2010.