Journal of Universal Computer Science, vol. 19, no. 13 (2013), 1940-1962
submitted: 7/11/12, accepted: 28/6/13, appeared: 1/7/13 © J.UCS

A Tool-based Semantic Framework for Security Requirements Specification

Olawande Daramola
(Department of Computer and Information Sciences
Covenant University, Ota, Nigeria
Olawande.daramola@covenantuniversity.edu.ng)

Guttorm Sindre
(Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU), Trondheim, Norway
guttors@idi.ntnu.no)

Thomas Moser
(Christian Doppler Laboratory for Software Engineering
Integration for Flexible Automation Systems
Vienna University of Technology, Vienna, Austria
thomas.moser@tuwien.ac.at)

Abstract: Attaining high quality in security requirements specification requires first-rate professional expertise, which is scarce. In fact, most organisations do not include core security experts in their software team. This scenario motivates the need for adequate tool support for security requirements specification so that the human requirements analyst can be assisted to specify security requirements of acceptable quality with minimum effort. This paper presents a tool-based semantic framework that uses ontology and requirements boilerplates to facilitate the formulation and specification of security requirements. A two-phased evaluation of the semantic framework suggests that it is usable, leads to reduction of effort, aids the quick discovery of hidden security threats, and improves the quality of security requirements.

Keywords: security requirements, ontology, requirements boilerplates, information extraction, security threat, misuse cases.
Categories: D.2.1, M.4, M.8

1 Introduction

The advent of the Internet and virtual environments that afford systems integration and collaboration among remotely based systems has increased the importance of security considerations when developing software systems. Security requirements specification entails the formal documentation of identified security needs of a system in order to ensure the development of secure information systems. However, attaining good quality in security requirements specification requires first-rate professional expertise, which is scarce. There is a lack of security experts or security requirements engineers in many organizations. This is because most requirements engineers have had no training in identifying, analysing, specifying, and managing security requirements (SR) [Firesmith 04; Salini 11]. Hence, when security requirements are

defined, they are often either too vague or overly specific in constraining designers to use particular mechanisms [Firesmith 04]. This scenario motivates the provision of tool-based support for security requirements specification. The tool-based support will: 1) assist the requirements analyst (subsequently referred to as analyst) to identify security threats, which is usually a manual procedure in which the quality of SR depends on the expertise of the human personnel; 2) stimulate the adoption of appropriate defence strategies to deal with the identified security threats; 3) enable the formulation of SR in a consistent way, minimizing ambiguity, and enhancing correctness of SR; and 4) reduce the effort needed for security requirements specification by allowing the reuse of previously specified SR in subsequent instances. The overriding aim is to complement the capabilities of the analyst in the task of security requirements specification so that the quality of SR is improved and the amount of human effort reduced.

In order to realize this aim, we have adopted a semantic-based approach that uses an integration of ontologies and requirements boilerplates to complement the activities of the analyst during security requirements specification. A requirement boilerplate is a predefined textual template that can guide the way requirements are written so that a consistent pattern can be maintained when writing structurally similar requirements [Hull 04]. The use of ontologies provides the necessary background knowledge, and domain knowledge that is required to identify security threats, and recommend appropriate countermeasures, while the requirements boilerplates provide a reusable template for writing SR in a consistent way in order to minimize ambiguity. Relative to previous approaches, the main contribution of this work is the provision of an elaborate semantic-based procedure that enables the tool-assisted formulation of SR in a way that enhances quality, and reduces effort needed to formulate SR. This is because our approach: 1) enables identification of security threats from security threat description scenarios; 2) provides recommendation of defence actions as countermeasure to identified security threats; and 3) enables pattern-based reuse of requirements boilerplates when writing SR.

The rest of this paper is as follows. Section 2 gives an overview of contextual background and related work, while Section 3 presents our approach. Section 4 discusses the evaluation, results and threats to validity. The paper is concluded in Section 5 with a discussion of further work.

2 Background and Related Work

In this section, we provide background information on security requirements engineering (SRE), application of ontologies to SRE, and the use of boilerplates for security requirements specification. Additionally, we discuss related work.

2.1 Security Requirements Engineering (SRE)

A requirement is "a condition or capability that must be met by the system to solve a problem or achieve an objective" [IEEE Std. 98]. Requirements engineering is a systematic process to elicit, analyse, specify, validate, and manage such requirements [Kotonya & Sommerville 98]. Security requirements engineering (SRE) can then be defined as requirements engineering specifically targeting the elicitation,

specification, analysis, validation and management of security requirements, where security can be understood as the ability of the system to cope with malicious attacks, ensuring confidentiality, integrity and availability of system data and functions. SRE is designed to ensure early consideration of probable security concerns when developing secure software systems. It aims to integrate the security needs of a system particularly from the attacker's perspective into the software development process as early as possible. According to [Chandrabrose 11], security requirements objectives can be categorized as authentication, authorization, integrity, intrusion detection, non-repudiation, confidentiality, and auditing. Some of the most well-known SRE approaches are described as follows.

i) *Comprehensive, Lightweight Application Security Process (CLASP)*
The CLASP approach [Viega 05; Graham 06] is a life-cycle process that suggests a number of different activities across the software development life cycle to improve security. The key steps of the CLASP approach are: 1) identify system roles and resources; 2) categorize resources into abstractions; 3) identify resource interactions through the lifetime of the system; 4) for each category, specify mechanisms for addressing each core security services. The CLASP handles security requirements by performing a structured walkthrough of resources, and determining how they address each core security service throughout the lifetime of that resource [Graham 06].

ii) *System Quality Requirements Engineering (SQUARE)*
SQUARE is a requirements engineering methodology for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications [Mead 05]. The process consists of nine steps performed in a sequential order by a team of requirements engineers, including at least one expert in risk assessment methods, and project stakeholders. The steps are: 1) agree on definitions; 2) identify assets and goals; 3) develop supporting artifacts; 4) perform risk assessment; 5) select elicitation technique; 6) elicit security requirements; 7) categorize requirements; 8) prioritize requirements; 9) inspect requirements. SQUARE is performed at the requirements elicitation stage of the development life cycle to develop security-related system requirements. By guiding stakeholders and requirements engineers through the specification of security requirements, SQUARE ensures that security is addressed early in the project life cycle in the same way as functional attributes and other quality attributes [Salini 11].

iii) *The Security Requirements Engineering Process (SREP)*
SREP [Mellado 07] is based partially on SQUARE but incorporates consideration of the Common Criteria [Common Criteria, 99] - an international standard (ISO/IEC 15408) for computer security certification - and notions of reuse. SREP is quite similar to SQUARE [Salini 11]. It is a nine-step process consisting of the following activities: 1) agree on definitions; 2) identify vulnerable and/or critical assets; 3) identify security objectives and dependencies; 4) identify threats and develop artifacts; 5) risk assessment; 6) elicit security requirements; 7) categorize and prioritize requirements; 8) requirements inspection; and 9) repository improvement.

iv) *Secure Tropos*
Secure Tropos [Mouratidis 04] is a methodology that is based on the i*-modelling framework for agent-oriented software development. It extends the Tropos methodology [Yu 01], by including concepts for modelling the security aspects of systems. The additional concepts introduced to Secure Tropos are: security constraint, secure dependency, and secure entity. Thus, the Secure Tropos process also extends the Tropos process with phases to analyze and model the new concepts. These activities produce different kinds of diagrams, which are used as input to the later activities. The activities are: security reference modelling, security constraint modelling, and secure capabilities modelling [Mouratidis 04].

v) The CORAS Method
CORAS [Braber, 07] is a UML-based method for model-based security analysis that is originally inspired by the notion of misuse cases. The misuse case (MUC) models the various ways in which the activities of a malicious user can be a threat to a system. It is consist of a diagram model – called MUC, and textual equivalent called the textual misuse case (TMUC). The MUC is an extension of the regular Use Case (UC) diagram with new additional concepts such as misusers, misuse cases and mitigation use cases in order to elicit the security requirements of a system [Sindre 05]. CORAS is a seven steps process that provides a customized language for threat and risk modeling and has a detailed guide on how the language should be used to capture and model relevant information during the security analysis. CORAS methods provides a computational tool designed to support documenting, maintaining and reporting analysis results through risk modeling, table-based documentation, consistency checking and more. The seven steps of CORAS are: 1) Introduction; 2) high level analysis; 3) approval of target description; 4) risk identification; 5) risk estimation; 6) risk evaluation, and 7) risk treatment.

2.2 Application of Ontologies to SRE

Ontology is a shared formal conceptualization of a domain that allows definition of semantic relationships between entities, and inference of knowledge through reasoning at run-time [Happel 10]. Ontologies have an important role to play in SRE. Research efforts on security ontologies such as [Fenz 09; Herzog 07; Kim 05] attest to this. In [Donner 03] the use of ontology was suggested as the solution to the problem of vaguely defined vocabularies among security practitioners.

According to [Souag 12], specific applications of ontology to SRE include:

(i) Security taxonomies – this are concept hierarchies of security terms and concepts.
(ii) General security Ontologies – this contains definition of all aspects of security such as assets, threats, and vulnerabilities. They are knowledge infrastructures that are created to support SRE activities such as threat modelling and security risk analysis.
(iii) Specific security ontologies – this have been developed to support specific security activities such as intrusion detection, computer network attack, and system vulnerabilities.
(iv) Security ontologies for Semantic Web – this have been developed for the security of semantic web resources such as software agents, and web services.

(v) Security ontologies for risk analysis – this are designed for the analysis of industrial risk that are associated with identified security threats.
(vi) Ontologies for security requirements – this supports the definition of security requirements.

Generally, a good ontology will facilitate more effective reporting of incidents, sharing of information, and interoperable security collaborations among different organizations. Our proposed framework uses ontology to ensure the standardization of vocabulary in security requirements specification, threat identification, and the recommendation of appropriate countermeasure to identified threats.

2.3 Boilerplates for Security Requirements

The notion of requirements boilerplates (RB) which stems originally from the work of [Hull 04], and subsequently applied in [Daramola 11] enables the writing of requirements in a consistent manner. A requirement boilerplate is a pre-defined structural template for writing requirement statements. It imposes a uniform structure on the way requirements are written, by affording a level of expressivity akin to using natural language, yet minimising ambiguity in requirement statements. The fixed parts of requirement boilerplate are reused when writing requirements, while the analyst can fill in the parameter parts manually.

An example of a boilerplate taken from the webpage[1] is:

"BP2: The <system> shall be able to <action> <entity>"

Here, *BP2* is the label of this particular boilerplate. The terms in < > brackets are parameters where something must be filled in when the boilerplate is instantiated to a concrete requirement. The words that are outside brackets are the fixed syntax elements (FSE) that will be kept as-is when the boilerplate is instantiated. An example of an instantiation of this particular boilerplate, would be *"The Internet banking system shall be able to authenticate all its users"*. In this case <*action*> has been replaced by "authenticate" and <*entity*> by "all its users". In some cases, several boilerplates may be combined to make precise and testable requirements, e.g. combining *BP2* with *BP37* **...at least <percentage> of the time** will yield the requirement *"The Internet banking system shall be able to authenticate all users at least 99.99% of the time"*.

Thus, the use of boilerplate will ensure that a unified structure and style of writing is used for requirements that pertain to specific classes of system function, capability, goals, or constraints. The FSE in each boilerplate will remain the same for all requirements that used a certain boilerplate. For instance, all who used BP2 + BP37 to specify that the system should be able to do something with some specific frequency, will now use phrases "shall be able to", "at least", "times per", rather than various other phrases that could have more or less the same meaning, e.g., "have the ability to", "be capable of", "a minimum of", "more than", "shots per", etc.

Therefore, using boilerplates will: i) facilitate the reuse of parts of the requirements text (viz. the FSE), as well as hints what should be filled in for parameters; (ii) help to attain better quality of requirements, e.g., BP35 reminding

[1] http://www.requirementsboilerplates.org

analysts that one should be precise about the required frequency of some action, thus encouraging more quantification of requirements, which is vital for testability (viz. using boilerplates as preliminary basis for requirements checking); and (iii) lead to completeness of the specification, e.g., looking at the boilerplates available, the analyst or domain expert might be reminded of requirements that they would otherwise have forgotten to specify (viz. using the boilerplates catalogue as a checklist).

Boilerplates could also be applied to security requirements (SR), and it is possible to integrate the formulated boilerplates with a security-related ontology for specifying SR. In [Firesmith 05], Firesmith provided a foundation for the mapping of specific security requirements to the types of defence actions to address them. He identified four different types of defence against security threats, which can be used to assign specific security threats to the types of defence actions to counter them. These are:

(i) Prevention of malicious harm, security incident, security threats and security risks.
(ii) Detection of malicious attack, security incidents, security threats, and security risks.
(iii) Reaction to detected malicious attack.
(iv) Adaptation of system to avoid or minimize the negative consequences of the malicious harm, security incidents, security threats and security risks. This could also be in terms of recovery of system from attacks.

For each of the defence types, Firesmith also gave specific examples.

The examples in [Firesmith 05], could be the basis for some generic SR boilerplates, e.g.

SecBP1: **The** <*system*> **should [prevent | detect] at least <***percentage***> of** <*harm* | *incident* | *threat* | *risk*>

SecBP2: **Upon detection of** <*harm* | *incident* | *threat* | *risk*> **the system shall** <*action*>

SecBP11: **...of attacks with maximum duration** <*number*> **<***time unit***>**

SecBP12: **...made by attackers with profile <***attacker profile***>**

SecBP21: **...at least <***[0-100]***>% of the time**

Here, *SecBP11*, *SecBP12*, and *SecBP21* are parts that could be optionally concatenated with *SecBP1* or *SecBP2* respectively.

The use of boilerplates for security requirements specification in practical terms will involve formulating requirement boilerplates for the different aspects of security such as authentication, authorization, integrity, intrusion detection, non-repudiation, confidentiality, and auditing. Therefore, more boilerplates could be formulated, both as core parts and as attachments, but since boilerplates for security will vary for different domains, we cannot go into more detail here. However, experienced personnel must create the boilerplates prior to security requirements specification as an upfront investment, while it should be updated periodically as new types of requirements emerge. By so doing, the boilerplate repository becomes an organisational asset for security requirements specification that can be useful when there is paucity of experienced security requirements analyst personnel.

2.4 Extending Boilerplates with Ontologies

It is possible to use boilerplates without any ontology, and the originators of boilerplates did not propose any ontology extension. Without ontologies, the boilerplates will still be useful in providing reuse of at least the FSE, and probably some quality increase by writing requirements in a fixed style, where one is reminded of the need to quantify requirements where possible. Extending the boilerplates with ontologies gives some extra advantages, such as allowing the validation of terms and relationships that are contained in requirements to ensure they are used correctly and acceptably in the domain concerned [Daramola 12]. For example, pertaining to the parameterized parts in italics in the boilerplate examples above (*SecBP1*), where the analysts themselves have to fill in numbers, single words or phrases in the boilerplate requirements, an ontology could be developed based on standards documents for that domain. The domain ontology will ensure that whenever the analyst fills in a word or phrase denoting some recognized concept from the ontology, say "ACC" (Automatic Cruise Controller) in the case of developing automotive software, the support tool could search for the concept ACC in the domain ontology, and - if finding it - know (i) whether there are any other terms which might be synonyms for this, and thus discover relationships between requirements that would not easily have been discovered otherwise, because of different terminology used; (ii) what other components are closely related to the ACC; and (iii) what various security levels is typically required of ACC, etc. Similarly, for other domains, e.g., aviation, one would have definitions of relevant concepts there, like flight, plane, pilot, runway, tower, etc. stored in a domain ontology. Hence, to put it simply, if just using boilerplates without ontologies, the analysts mainly get help with reusing the FSE (bold phrases) of examples like those above, but are left on their own with the parameters (italics) to fill the brackets. With ontologies there is help also with ensuring consistent use of language, discovering requirements that are related to each other because they address the same or closely related concepts or procedures in the system, etc. Hence, a tool platform that enables the leveraging of domain knowledge via an ontology, and use of boilerplates will be potentially useful to a requirements analyst in the process of security requirements specification.

2.5 Related Work

We shall approach our review of related work from two perspectives. First we shall consider the recent approaches of ontology-based frameworks for security and second, the aspect of tool support for security requirements engineering (SRE).

Recent approaches of ontology-based frameworks for security include [Lasheras et al., 2009], where an ontology framework for reusing security requirements during requirements specification was presented. The work involved the creation of a risk analysis ontology and requirement ontology which are combined to represent reusable security requirements, and improve the detection of incomplete and inconsistent requirements, and also semantic processing in requirements analysis. It was an extensible framework that provided basis for a lightweight method to elicit, and specify security requirements, based on security standards. The framework is typically a combination of ontologies that will be useful to ontology experts who are not experienced in security issues. However, the work was not focussed on detecting

security threats from textual sources as done in our own approach, neither was it a tool-based framework. [Chikh et al., 2011] proposed a framework for information security requirements (ISRs) using ontologies. The framework enables ISRs traceability and reuse based on three kinds of generic ontologies – software requirement ontology, application domain ontology, and information security ontology. The work proposed the design of an ontology-based information security requirements engineering framework, which supports analysts in building and managing their ISRs by leveraging the three ontologies. The authors anticipated that the proposal will help requirements engineers to create and understand the ISRs. The work was strictly a proposal that did not include any implementation or evaluation. [Massacci et al., 2011] presented an amalgamated and extended ontology for modelling security requirements. The extended ontology integrates the existing concepts from the Problem Frames and Secure i* methodologies in order to realize a more complete basis for modelling security requirements due to missing constructs in the individual approaches. The expressiveness of the proposed extended ontology was tested by modelling the security requirements of a case study from the Air Traffic Management domain. Similarly, [Blanco et al., 2011] after conducting a systematic review of existing proposals, proposed a basis for an integrated and united security ontology since existing security ontologies seem to focus on specific domains. They identified three key requirements that an integrated security ontology should consider, which are static knowledge - which will allow the concepts collected in the ontology to be properly identified; dynamic knowledge – which will ensure that the knowledge collected in the ontology can be used to infer other knowledge; and reusability - which will ensure the fact that the ontology is developed by taking into consideration aspects that permit its reuse and shareability. The tool-based orientation of our approach is a deviation from existing ontology-based frameworks for security, in that it targets real-time support for the requirements analyst during security requirements specification. It leverages ontology and a template-based approach –boilerplates - for both the elicitation and formulation of security requirements from textual sources in order to reduce effort and enhance quality.

In terms of tool support for SRE, we shall classify tool for SRE into two broad categories – front-end tools and back-end tools. Front-end tools and approaches are those that facilitate the elicitation, modelling, and analysis of security threats in order to derive SR, while the back-end tools are those that help with the specification and validation of SR, and their integration with other requirements. Notable examples of front-end tools include: SecTro [Pavlidis 12], - a CASE tool that supports automated modelling and analysis of security requirements based on the Secure Tropos approach. The ST-Tool [Giorgini 05] supports the Secure Tropos methodology. Its main goals are to support the translation of Secure Tropos models into formal specifications, and serve as a front-end tool for formal analysis of Secure Tropos models. The jMUCMNav editor (Java Use Case Map Navigator), [Bizhanzadeh 11] is a modelling tool for Misuse Case Maps (MUCMs) in designing secure architectures for business processes. jUCMNav simply focuses on modelling for use case maps (UCM) and supports all UCM notations. Other front-end SRE tools that are worth mentioning are SeaMonster [Tøndel 10; SeaMonster 07], and Suraksha security workbench [Maurya 09].

Currently, there are more front-end tools than back-end tools for SRE. The SQUARE tool [Mead 05] is a back-end managerial tool that is designed to increase the quality of SRE process for the adopters of the SQUARE methodology. It supports core SRE aspects such as definitions, searching, and addition of new terms, identification of security goals, assets and privacy goals, performing risk assessment, identifying threats, prioritizing requirements, traceability, and exporting of requirements to other requirements management tool. Similarly, the prototype tool - *ReqSec tool* – that we have developed is an eclipse-based back-end tool that supports security requirements specification, and enables the integration of SR with other types of requirements. The unique feature of the ReqSec tool compared to other SRE back-end tools stems from its capability to facilitate automatic analysis of natural language requirements in order to assist the analyst during security requirements specification. It represents a first attempt to use semantic-based procedures for supporting both security threat identification and security requirements specification. In the wider requirements engineering context, approaches such as [Gleich 10] – ambiguity detection-, [Wilson 97; Fabrini 01] – requirement quality assessment-, are also based on natural language (NL) text analysis but did not use ontologies. The DODT [Farfeleder 11] tool does not have a focus for SRE, but it bears similarity with our approach, because it combines the use of ontologies and boilerplates to enable semi-automatic transformation of NL requirements into boilerplate requirements. However, it can only ensure the correctness of requirements based on the underlying domain ontology, and the writing of boilerplate requirements. Our approach does more, in that it entails the discovery of latent security threats contained in NL descriptions, and the recommendation of probable defence actions that aids the formulation of semi-formal boilerplate SR. Hence, the novelty of our approach is the provision of a backend tool for SRE that will minimize effort needed for security requirements specification, and offer a credible starting point for security requirements specification, particularly in cases where there is paucity of experienced personnel.

3 Overview of the Semantic Approach

A high-level schematic overview of our semantic approach is presented in Figure 1. The process starts with input of description of the security threat scenario, which should be represented as a textual Misuse Case (TMUC) [Sindre, 05]. This is followed by identification of type of attack and required defence action through semantic text analysis of the TMUC, thereafter suggestion of boilerplates to be used to the analyst, and finally specification of security requirements by the analyst. In sequel sections, we present a typical example of the description of a security threat described using a TMUC, and the description of the different activity steps of our semantic framework.

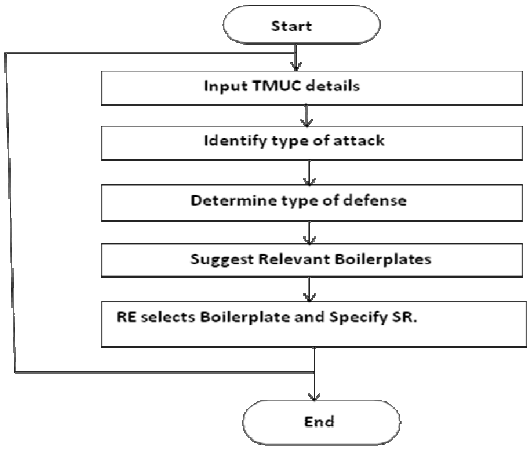

Figure 1: Activity Workflow of the tool-assisted SR Specification Process

3.1 Database Tampering - Example

In order to demonstrate how our tool-based framework can be applied, we hereby consider the example of a security threat description of database tampering scenario. The detail of the scenario is presented in Table 1 using a TMUC template.

Code: QC1	
Misuse Case Title	Tamper with database by web query manipulation
Name of System	Web Query System
Summary	A crook manipulates the web query, submitted from a search form, to update or delete information, or to reveal confidential information;
Basic Path	The crook provides some values on a product search form and submits. The system displays the product(s) matching the query. The crook alters the submitted URL, introducing a query error, and resubmits. The query fails and the system displays the database error message to the crook, revealing more about the database structure. The crook alters the query further, for instance adding a nested query to reveal secret data or update or delete data, and submits. The system executes the altered query, changing the database or revealing content that should have been secret.
Alternative Path	The crook does not alter the URL in the address window, but introduces errors or nested queries directly into form input fields.

Table 1: TMUC for Database Tampering Case

3.2 Input TMC details

The TMUC template [Sindre, 05] has two core aspects namely the basic path, and the alternative path. The basic path describes the security threat scenario that could be used by an attacker to cause harm to a system, while the alternative path specifies the other options that may be explored by an attacker or user with malicious intent. These two aspects together with the TMUC summary provide the key inputs used to identify the type of attack, and required defence for the system.

3.3 Identify Type of Attack

We used information extraction techniques to identify the type of attack described by a TMUC template. The textual input was semantically analysed in order to identify and extract the theme words. A theme word can be the subject of a sentence (noun), or an action word (verbs) or a word collocation that connote a security threat to a system. The knowledge that is captured in the definition of basic threat ontology (BTO) (see Figure 2), WordNet ontology, and the domain ontology (DO) are used to facilitate the identification of theme words. Core procedures for semantic analysis of natural language text such as tokenization, parts-of-speech tagging, syntax parsing, morphological analysis, and ontology-based inferencing were implemented by using algorithms that based on the Stanford NLP toolkit[2], Word Net java API, and the Jena semantic framework[3].

3.4 Determine the Type of Defence using the Basic Threat Ontology (BTO)

The BTO contains a mapping of different kinds of security threats to specific defence actions based on information that was gathered from the literature and a number of existing security ontologies. The BTO is a major investment and a core knowledge infrastructure of the framework. The defence actions are the ones proposed by Firesmith in [Firesmith 05]. We reused all the essential aspects of the threat description in Security Ontology [Herzog 07] as foundation for developing the BTO, which included some additional concepts. The BTO has a total of 98 classes, 46 restrictions and 9 object properties. The key object properties include *hasDefense* – which associates a threat with a specific defensive action, *hasThreat* – which associate a threat with an asset, *isThreathenedBy* – inverse of hasThreat, *isThreatTo*, *isSameAs*, - which describes equivalent concepts. Each security threat in the BTO was mapped to one or more defence actions (viz. detect, prevent, adapt, react, recover) using the *hasDefense* object property. Figure 2, presents a view of the BTO illustrating how specific types of attacks have been mapped to corresponding defence actions. The knowledge contained in the BTO is used for automatic recommendation of appropriate defence actions when a particular type of attack has been identified from the TMUC input details. The Pellet OWL Descriptive Logics (DL) reasoner was used as the ontology reasoning engine for the BTO.

[2] http://nlp.stanford.edu/software/lex-parser.shtml
[3] http://jena.sourceforge.net/

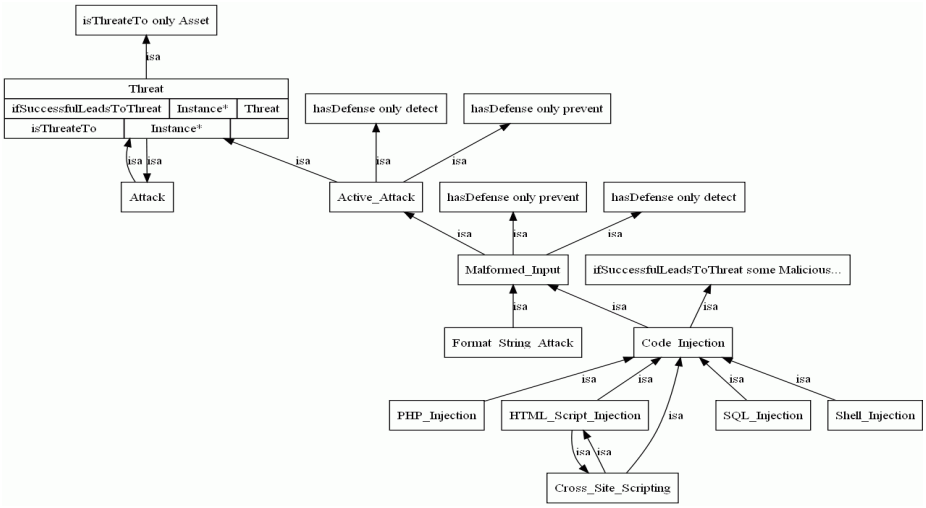

Figure 2: A view of the description of Malformed Input threat in the BTO using OntoViz

3.5 Suggestion of Relevant Boilerplates

The output of the information extraction process is a list of recommendations comprising a set of defence-action and attack-type pairs R (see Figure 3) such that:

R= $\{r_1, r_2, ..., r_n\}$ i.e. $r_i = <d, t>$

Here *d* is a defence-action and *t* is a specific type of attack that has been identified. For example:

<detect> <eavesdropping>
<prevent> <code Injection>
... etc.

The recommended pairs are derived by combining the relevant theme words that were extracted from a particular TMUC description and specific attack types (threat concepts) that are described in the BTO. A theme word will be considered relevant to a particular security threat scenario if any of the following rules are true [Daramola 12]:

(i) if the theme word syntactically matches an existing BTO threat concept;
(ii) if the theme word in its root form (lemma) can be associated with a BTO threat concept;
(iii) if the theme word is either a synonym, hyponym, or hypernym of a BTO threat concepts; and
(iv) if a BTO threat concepts is a sub-concept of a concept already identified by any of (i) - (iii).

In cases when the theme word is not a single word (e.g. phrase), the *head word* – that which is most significant to the meaning – is selected and used in the inference rules. The corresponding defence action for an identified type of attack is inferred by exploring the set of pre-defined mappings of specific threats to defence actions that are specified in the BTO. The analysis of theme words to determine their different word forms is enabled by the using the WordNet ontology and the WordNet Java API,

while the Pellet OWL Descriptive Logic reasoner was used to facilitate ontology reasoning on the axioms and concepts of the BTO.

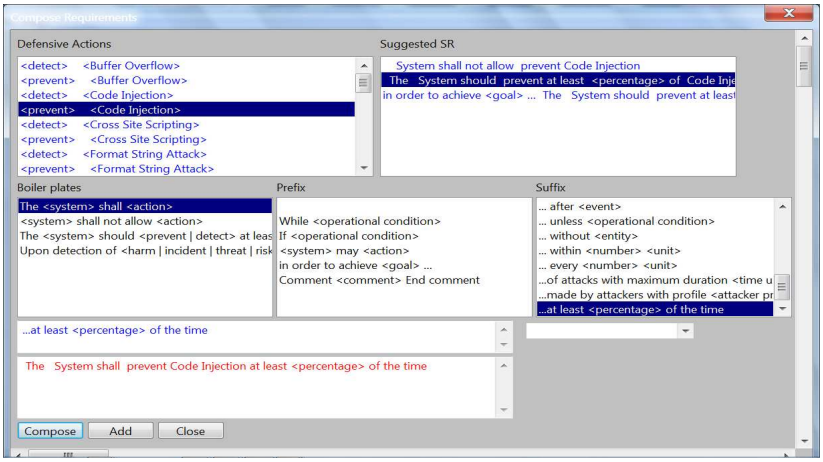

Figure 3: A snapshot of suggestions for database tampering from the tool

3.6 Formulation of Security Requirements and Reuse of Boilerplates for Security Requirements Specification

Beside the suggestion of boilerplates, the semantic framework aids the requirements analyst during the formulation of SR by facilitating the reuse of existing boilerplates that are been used previously. The activities of tool-assisted SR formulation and reuse of boilerplate for security requirements specification are described as follows.

3.6.1 SR Formulation

At the start of the security requirements specification process, when no requirements have been specified, the analyst will have to select a specific <defence-action, attack-type> option from the list of recommendations, and from the boilerplate repository, must select a core boilerplate, and optionally select, a prefix, or suffix or both. For example, when the analyst, selects *SecBP1*(core boilerplate), and *SecBP21* (suffix) [see Section 2.3] and highlights a recommended <defence-action, attack-type> option say "<prevent, Code Injection>", the resulting boilerplate SR would have keywords "prevent" and "Code Injection" already inserted in the right places as below:

The <system> should prevent at least <[0-100]>% of Code Injection at least < [0-100]>% of the time

For each SR formulation, the fixed syntax elements (FSE) in the selected boilerplate, prefix, and suffix are reused, while the selected <defence-action, attack-type> substitutes the <action> placeholder in the selected boilerplate. The analyst can then fill in the remaining part – marked with "< >" - of the boilerplate requirements that require specific data to complete the SR formulation. While completing the remaining part, data that can be inferred from knowledge contained in the underlying DO e.g. the specific system name for <*system*> are also suggested to the analyst. In this way, the analyst can save some effort, attain consistency, completeness, and

correct use of domain vocabulary, which ultimately improves the quality of specified SR.

3.6.2 Reuse of Requirement Boilerplates for SR

The tool is able to learn as the analyst selects specific requirement boilerplates and uses them to formulate SR. It keeps track of the frequency of use of each requirement boilerplate and maintains an updated count of each one. When the analyst has completed the formulation of few SR (two or more), the relevant requirement boilerplate patterns that tend to be more used are displayed in a ranked order as probable candidates to be reused for a new SR formulation [Daramola 12]. The search process for relevant requirement boilerplates to reuse entails looking at all previously used requirements boilerplate patterns that contains the keyword specified in the highlighted recommended <defence action, <attack type> option and ranking them based on their frequency of usage. For example if the recommended option "prevent Code Injection" is highlighted by the analyst, a ranked list of auto-generated requirements boilerplate patterns – combinations of boilerplate core parts, prefix, and suffix, that contains the keywords "prevent" that have been previously used are displayed as candidates to reuse in a new SR formulation. With this, the analyst is able to find reusable requirement boilerplates quicker, and avoids having to look through the complete set of boilerplates – core parts, prefix, and suffix in the boilerplate repository before composing a SR. This way, the analyst can save some effort by using the tool to specify SR.

4 Evaluation

In order to assess the semantic framework, we have conducted evaluation in two phases. In the first phase, we assessed the usability of the tool-based semantic framework to support security requirements specification. We conducted a preliminary evaluation by using a controlled experiment with seven subjects. The subjects were Master degree students of software engineering of NTNU, Norway, who volunteered to participate in the experiment. The participants were paid for taking part in the experiment. The second phase of evaluation focussed on assessing the quality of security requirements that were specified by participants in the first evaluation. To do this, we requested experienced requirements analysts who have ample experience of working with security requirements to assess the quality of the SR specified by the participants of the first evaluation that used the tool-based semantic framework.

4.1 First Evaluation

4.1.1 Background of Participants

An assessment of the background of participants through a pre-experiment questionnaire revealed that the participants had good background knowledge in the specific areas such as system security, requirements engineering, ontology, and boilerplates that pertain to the experiment. They have all taken two relevant courses – *TDT4237 Software security* and *TDT4242 Requirements engineering and testing* at

the NTNU, Norway. In addition, the majority of the subjects also claimed to have some industrial work experience, although the majority only as part time positions or summer internships while studying.

4.1.2 Evaluation Procedure

The participants were asked to use the *Reqsec tool*[4] during a controlled experiment that lasted for 1.5 hours. They were presented with four security threat scenarios – TMUC- in four different domains. The TMUC deals with: (i) Air Traffic Control System – fabrication of false clearance; (ii) Electronic Passport System – identity theft of card holder; (iii) Network Security Bridge – hacking of network host computer; (iv) Database Tampering - manipulation of web query. All the participants performed the same task at any given time during the experiment. The participants were given a five minutes tutorial on the use of the tool[5] before they commenced the experiment. They were required to assess the tool along six dimensions - *perceived usefulness (PU), perceived ease of use (PEOU), intention to use (ITU), reuse (reu), accuracy (acc), and serendipity (sere)* - through a post-experiment questionnaire that was based on a 5-point Likert scale. The mean scores out of a maximum of 5.0 for each of the six dimensions are shown in Table 1.

4.1.3 Results

The analysis of the results from the post-experiment questionnaire revealed that the tool had its highest mean rating in the aspects of *perceived ease of use (PU)*, and *serendipity (sere)* – the users acknowledged that the tool offered suggestions that they had not thought about originally. The tool also had good rating in other aspects such as *reuse*, *accuracy*, and *intention to use*. All the participants stated emphatically that they would use the tool.

In the free comments feedback section of the questionnaire, the participants revealed a positive general perception of the tool as potentially viable to support security requirements specification, and admitted their willingness to use it. Most agreed that the tool is easy to use, and capable of assisting the analyst. A few of them were particularly impressed that the tool enabled them to write security requirements that they did not think about initially until when they saw the suggestions from the tool. They all agreed that although the tool offers useful support for security requirements specification, it cannot be solely relied upon. This is because there were occasions when the tool failed to suggest certain expected options. Some of them advised that the tool would perform better if the quality of the underlying ontology is improved. They also mentioned a number of areas that should be improved in the tool. This includes the fact that 1) the tool's interface did not scale well on the MacOs systems compared to Windows; and 2) the need to be able to save the requirements that pertain to a TMUC all at once in the repository and not one at a time. We agree with the observations of the participants and would seek to revise the subsequent version of the tool based on the observations by participants.

Generally, the result of the first evaluation demonstrates the potential of the tool to first, simplify, and significantly aid the analyst during the security requirements specification process, particularly when the analyst is not highly experienced.

[4] https://www.idi.ntnu.no/~wande

[5] https://www.idi.ntnu.no/~wande/Guide_for_Reqsec_Tool.htm

Second, facilitate a reduction in the effort expended on security requirements specification, particularly as the process progresses. Third, ensure that correct terms of the domain are used when formulating SR, and in a consistent way, thus reducing ambiguity. However, our inspection of the specified security requirements revealed consistency in the use of language and pattern of expression in formulated SR that pertain to same security threat scenario by different individuals, which is mainly due to the use of boilerplates and ontologies.

Metric	Mean	Std
PU	3	0.433013
PEOU	3.714286	0.698638
ITU	3.357143	0.481039
Reu	3.214286	0.393398
Acc	3.285714	0.95119
Sere	4	0.816497

Table 1: Mean score rating for Tool Assessment

4.2 Expert Assessment of Quality of SR

The second phase of the evaluation involved the use of experienced requirements analysts to assess the quality of the SR specified by the subjects used for the first phase of evaluation. The assessment was based on ten desirable requirements quality metrics [Stokes, 1991; IEEE Std. 98], which are defined as follows [Wilson 97]:

i) *Complete* - the requirements specification precisely define all real world situations that will be encountered, and the capability to respond to them.

ii) *Consistent* - there is no conflict between individual requirement statements, that define the behaviour of essential capabilities; and specified behavioural properties and constraints do not have a negative impact on that behaviour.

iii) *Correct* - the requirements specification accurately and precisely identify the individual conditions and limitations of all situations that the desired capability will encounter, and it defines what should be the proper response to those situations.

iv) *Modifiable* - the requirements specifications is able to identify related concerns and grouped them together, while the unrelated concerns have been separated.

Testable - the requirements are stated precisely, such that a pass/fail or quantitative assessment criteria can be derived to validate their correct Implementation.

Traceable - each statement of requirements has a unique identification that makes it easy to trace them subsequently.

v) *Unambiguous*: the requirement statements are not subject to multiple interpretations.

vi) *Understandable*: the language used to specify the requirements are simple, concise and easy to understand.

vii) *Valid* - all the stakeholders would be able to understand, analyze and accept or approve the requirements
viii) *Verifiable* - there is high possibility that requirement specifications at the current level of abstraction will be consistent with those at another level of abstraction

4.2.1 Evaluation Procedure

We collated the requirements specified by the seven participants for the four problem scenarios during the first evaluation into four single lists of merged requirements – a list for each problem scenario - ensuring that cases of exactly duplicated requirements do not exist. The summary of the collated requirements is shown in Table 2. The experienced analysts were then requested to assess the quality of the four sets of requirements along the ten requirements quality dimensions using a questionnaire that is based on the 5-point Likert scale. The mean score out of a maximum of 5.0 for each of the ten dimensions for the four sets of requirements (R1-R4) are shown in Tables 3 and 4.

	Security Problem Scenario	**Description**	**Total number of merged requirements**
1.	Automatic Traffic Control System (ATC) – R1	Fabrication of false clearance	12
2.	Electronic Passport System (EPS) – R2	Identity theft of card owner	12
3.	Network Security Bridge (NSB) – R3	Hacking of network host system	26
4.	Database Tampering (DAT) – R4	Manipulation of web query.	19

Table 2: Summary of Collated Requirements

4.2.2 Results

The outcome of the assessment of the quality of SR by the experts revealed that the specified SR had above average rating in terms of their *completeness*, *consistency*, *modifiability*, *testability*, *traceability*, *lack of ambiguity*, *understandability, and validity* attributes. The SR got relatively higher rating in terms of *completeness* and *consistency*. The SR got below average ratings in the aspect of *correctness*, and *verifiability*. In the free comments feedback section of the questionnaire, the experts also reviewed many of the specified SR to make them less ambiguous and more understandable.

	R1: ATC		R2: EPS		R3: NSB		R4: DAT	
	mean	std	mean	std	mean	std	mean	std
Complete	3	1	3	1	3.3	1.155	3	1
Consistent	4	1	4	1	4	1	3.67	1.155
Correct	2.3	0.577	2	1	2.67	0.577	2.67	0.577
Modifiable	2.67	0.577	2.67	0.577	3.3	0.577	3	1
Testable	2.67	2.081	2.67	2.081	3.3	1.527	3	1.732
Traceable	4.3	1.155	4	1.732	4.3	1.155	4	1.732
Unambiguous	2.33	1.155	2.3	0.577	3	1	2.67	0.577
Understandable	3	1.732	2.3	1.527	2.3	1.527	3	1.732
Valid	2.67	1.155	2.67	1.155	2.3	1.5275	2.67	1.155
Verifiable	2.3	1.155	2	1	2	1	2.3	1

Table 3: Mean score rating of Specified SR quality by Experts

	Mean R1- R4
Complete	**3.075**
Consistent	**3.917**
Correct	**2.41**
Modifiable	**2.91**
Testable	**2.91**
Traceable	**4.15**
Unambiguous	**2.58**
Understandable	**2.65**
Valid	**2.58**
Verifiable	**2.15**

Table 4: Global Mean score rating of Specified SR quality by Experts

The result of the second evaluation suggests that the tool-based semantic framework is not only usable to support security requirements specification but can also facilitate improvement in many of aspects of security requirements quality. Generally, the result confirms the potential relevance of the tool-based framework to complement an inexperienced analyst during security requirements specification. We can also conjecture that the low rating of specified SR in terms of *correctness*, and *verifiability* is due more to the competence level of the subjects used in the experiment and not necessarily due to a deficiency in the tool-based framework. For example, we observed that although most of the SR that were specified by participants by using the tool were syntactically and semantically correct as far as the

domain vocabulary is concerned, the experienced requirements analysts still gave a low rating to many of the specified SR in terms of *correctness*. This means there are other implicit considerations for correctness known to the experts that were not obvious to the less experienced participants. Hence, there is a probability that a more experienced analyst would be able to specify SR of higher quality in terms of *correctness*, and *verifiability* by using the tool.

4.2 Lessons Learned

Our experience from the first phase of evaluation emphasised the need for high quality underlying ontologies – BTO, DO - and the boilerplate repository. Hence, an upfront and crucial investment is to ensure that good quality BTO, DO and a rich boilerplate repository are available at the onset of the tool. In order to cater to this, the tool comes pre-loaded with the BTO and the boilerplate repository as basic artefacts, while a DO can be imported into the tool. Also, provision was made to ensure the evolution of the BTO, DO, and boilerplate repository with time. To do this, we have made it possible to continually revise the ontologies BTO, DO, and boilerplate repository from within the tool's environment. The tool includes an ontology management module that allows the addition of new concepts, properties, and axioms to an existing ontology, while the boilerplate management module allows the boilerplate repository to be updated. Thus, the tool can be customised, and adapted to cater for future emerging requirements.

Also, from the second phase of evaluation, it was evident that although the tool-based framework offer a good starting point for specifying SR, inspection of specified SR by more experienced personnel will be crucial to further improve quality of SR. Hence, the tool-based framework will provide optimal value when integrated into existing security requirements engineering (SRE) frameworks.

4.3 Evaluation Threats

Ordinarily, an industrial case study would give a different perspective to the evaluation of the tool and the quality of tool support. However, the subjects used for the controlled experiment during the first phase of evaluation are sufficiently knowledgeable in the relevant areas such as requirements engineering, system security, ontologies, and requirements boilerplates having taken taught courses in these areas. This makes them suitable as reasonable substitutes for real experts in a preliminary evaluation [Runeson 03]. Also, the evaluation was performed with only seven users, but although the statistical significance is reduced, the results are indicative of the acceptance of the approach evaluated. Moreover, our objective is to assess the potential usability of the tool to support security requirements specification. Evidence in literature suggests that a minimum of 5 subjects are sufficient to get a valid opinion on the usability of a tool [27].

Also, a concern could be that if the second phase of evaluation had been performed with a more number of experts, who have a more diverse background, the result could be different. We consider that the three experts used for the second evaluation, have the required experience and competence to give an unbiased and valid opinion on the quality of SR. The involved persons have a minimum of doctoral degree in software engineering, with diverse background in security requirements engineering research and practices. They also belong to different research institutions

and from two countries. So, while we currently see no reason to invalidate the observations of the three experts used to evaluate the quality of specified SR, an interesting point for further research is to have a wider group of experts to assess the quality of SR specified by the tool.

Another perspective to the evaluation could be to evaluate the tool alongside other requirements management tools (e.g. Doors) not specifically focussed on security, and then see which groups are able to come up with the most and best security requirements for a certain problem within a given time limit. Another alternative is to compare the performance of two groups of users – one using the tool to formulate requirements, and the other group using largely a manual approach to document requirements, but supported by MS Excel. Many in the industry use Excel to document requirements where the human has to fully write the requirements from scratch, and Excel used only to support plain editing, storage, retrieval, and possibly automated numbering of requirements. Either of this form of evaluation could also lead to a different result compared to what we have reported. However, comparative evaluation with other tools is not attractive as at now because, hardly could we find any other tool that have the same focus, and is set out to do exactly a similar thing as we envisioned. The option to compare the tool's performance relative to using MS Excel by two user groups is a possibility for the future, after this preliminary evaluation.

5 Conclusion

In this paper, we have presented the notion of semantic-based support for security requirements specification. Our approach employs a tool-based framework that uses a combination of ontologies and boilerplates to aid a requirements analyst in the process of security threat identification and eventual formulation of quality SR. It provides the attendant benefits of reducing the effort needed for the security requirements specification process, and offers a good starting point in cases when sufficiently experienced requirements analyst may not be available. The preliminary evaluation of the approach confirms that it is viable and usable for supporting security requirements specification, and that the specified SR have a generally acceptable rating in most aspects of the ten requirements quality dimensions used for evaluation. In future work, we will conduct a more elaborate evaluation by using industrial case studies to further validate the approach. Also, we shall seek means to further improve the performance of the tool, and extend the concepts to the aspect of safety.

Acknowledgement

The Norwegian Research Council through the *ReqSec* project, Norway, has supported this work while the first author of this paper was a Research Scientist at NTNU, Norway. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria.

References

[Bizhanzadeh 11] Bizhanzadeh, Y. and Karpati, P.: "jMUCMNav: an Editor for Misuse Case Maps", First Int. Workshop on Alignment of Business Process and Security Modelling (ABPSM'11), Riga, Latvia, (2011)

[Braber, 07] Braber, F., Hogganvik, M., Lund, Stølen, S., and Vraalsen, F: "Model-based security analysis in seven steps – a guided tour to the CORAS method", BT Technology Journal, 25(1):101-117, (2007)

[Chandrabrose 11] Chandrabrose, A. and Alagarsami: "Security Requirements Engineering – A Strategic Approach", International Journal of Computer Applications, 13, 3 (2011), 25-32

[Common Criteria, 99] Common Criteria Implementation Board, "Common Criteria for Information Technology Security Evaluation", Part 2: Security Functional Requirements. (1999).

[Daramola 11] Daramola, O., Stålhane, T., Sindre, G., Omoronyia, I.: "Enabling Hazard Identification from Requirements and Reuse-Oriented HAZOP Analysis", In: Proceeding of 4th International Workshop on Managing Requirements Knowledge, IEEE Press, 3-11 (2011)

[Daramola 12] Daramola, O., Sindre, G., Stålhane, T.: "Pattern-based Security Requirements Specification Using Boilerplates and Ontologies", Proceedings of Second International Workshop on Requirements Patterns (RePa '12), IEEE Press, (2012), 54-59.

[Donner 03] Donner, M.: "Toward a Security Ontology", IEEE Security and Privacy, (2003).

[Fabrini 01] Fabrini, F., Fussani, M., Gnesi, S., Lami, G.: "An Automatic Quality Evaluation for Natural Language Requirements", In Proceeding of the Seventh International Workshop on Requirements Engineering Foundation for Software REFSQ '01, Interlaken, Switzerland, 150-164 (2001)

[Farfeleder 11] Farfeleder, S., Moser, T., Krall, A., Stålhane, T., Zojer, H., Panis, C.: DODT: Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development, In proceedings of 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2011), (2011), 1 – 4.

[Fenz 09] Fenz S., Ekelhart A.: "Formalizing Information Security Knowledge", In 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09), (2009), 183-194.

[Firesmith 04] Firesmith, D.: "Specifying Reusable Security Requirements", Journal of Object Technology, 3, 1 (2004), 61-75.

[Firesmith, 05] Firesmith, D.: "A Taxonomy of Security-Related Requirements", Proceedings of the International Workshop on High Assurance Systems (RHAS'05), Paris, France, (2005)

[Giorgini 05] Giorgini, P., Massacci, P., Mylopoulos, F., Siena, J., Zannone, N.: "ST-Tool: A CASE Tool for Modeling and Analyzing Trust Requirements", Lecture Notes in Computer Science, Vol. 3477, (2005) 415-419.

[Gleich 10] Gleich, B., Creighton, O., and Kof, L.: "Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources", In Wieringa, R., Pearson, A (eds.) REFSQ '10, LNCS, vol. 6182, (2010), 218-232.

[Graham 06] Graham, D.: "Introduction to the CLASP Process", Build Security In, (2006). https://buildsecurityin.us-cert.gov/daisy/bsi/articles/ best-practices/requirements/548.html.

[Happel 10] Happel, H., Maalej, W., Seedorf, S.: "Applications of Ontologies in Collaborative Software Development", In Mistrík et al. (eds.): Collaborative Software Engineering. Springer, (2010).

[Herzog 07] Herzog A., Shahmehri N., and Duma C.: "An Ontology of Information Security", International Journal of Information Security, 1, 4, (2007), 1-23.
[Hull 04] Hull, E., Jackson, K., Dick, J.: "Requirements Engineering", Springer (2004).

[IEEE std. 98] IEEE Standard 830-1998, "Recommended Practice for Software Requirements Specifications", December 2, (1998).

[Kim 05] Kim, A., Luo J., and Kang M.: "Security Ontology for Annotating Resources", In 4th International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE'05) (2005).

[Kontoya, 98] Kotonya, G., Sommerville, I.: "Requirements Engineering: Processes and techniques", John Wiley & Sons, 1998.

[Maurya 09] Maurya, S., Jangam, E., Talukder, M., Pais, A.R. Suraksha: A security designers' workbench. Proc. Hack. (2009), 59–66.

[Mead 05] Mead, N., Stehney, T.: "Security quality requirements engineering (SQUARE) methodology", Proceedings of International Conf. on Software Engineering for Secure Systems (SESS'05), (2005), 1-5.

[Mellado 07] Mellado, D.; Fernandez-Medina, E.; & Piattini, M. "A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems." Computer Standards & Interfaces 29, 2 (2007): 244-253.

[Mouratidis 04] Mouratidis, H., Giorgini, P.: "Secure Tropos: A security-oriented extension of the Tropos methodology", International Journal of Software Engineering and Knowledge Engineering 17, 2 (2004), 285-309.

[Nielsen, 93] Nielsen, J, and Landauer, T.: "A mathematical model of the finding of usability problems", Proceedings of ACM INTERCHI'93 Conference, (1993), 206-213.

[Pavlidis 12] Pavlidis, M., Islam, S. and Mouratidis, H.: "A CASE tool to support automated modelling and analysis of security requirements", Lecture Notes in Business Information Processing, Vol. 107, (2012), 95-109.

[Runeson 03] Runeson, P.: "Using Students as Experiment Subjects – An Analysis on Graduate and Freshmen Student Data", In: Linkman, S. (ed.) 7th International Conference on Empirical Assessment & Evaluation in Software Engineering (EASE'03), (2003), 95–102.

[Salini 11] Salini, P. and Kanmani, S.: "A Survey on Security Requirements Engineering", International Journal of Reviews in Computing, 8 (2011), 1-8.

[SeaMonster 07] SeaMonster: "Security Modelling Software" (2007); Retrieved at: http://sourceforge.net/apps/mediawiki/seamonster/

[Sindre, 05] Sindre, G., and Opdahl, A.: "Eliciting Security Requirements with Misuse Cases", In: Requirements Engineering, vol. 10(1), (2005), 34 - 44.

[Souag 12] Souag, A., Salinesi, C., Wattiau, I.: "Ontologies for Security Requirements: A Literature Survey and Classification", In Bajec, M. and Eder, J (eds.) Advanced Information Systems Engineering Workshops, Lecture Notes in Business Information Processing (LNBIP), 112, (2012), 61-69.

[Stokes 91] Stokes, D.: "Requirements Analysis", Computer Weekly Software Engineer's Reference Book, (1991), 16/3-16/21. – Requirements Quality.

[Tøndel 10] Tøndel, I.A., Jensen, J., Røstad, L.: "Combining misuse cases with attack trees and security activity models", Proc. ARES'10, (2010) 438-445.

[Viega 05] Viega J. "The CLASP Application Security Process". Vol. 1.1. Training Manual. Secure Software Inc. (2005).

[Wilson 97] Wilson, W., Rosenberg, L., Hyatt, L.: "Automated Analysis of Requirement Specifications", In Proceedings of the International Conference on Software Engineering (ICSE '97), (1997), 161-171.

[Yu 01] Yu, E.: "Agent-Oriented Modelling: Software Versus the World", Agent-oriented software engineering AOSE-2001 Workshop Proceedings, LNCS 2222, (2001), 206-225.