Indonesian Journal of Electrical Engineering and Computer Science
Vol. 15, No. 3, September 2019, pp. 1578~1585
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v15.i3.pp1578-1585

A novel intelligent model for classifying and evaluating non-functional security requirements form scenarios

Akram AbdelQader
Department of Software Engineering, Al-Zaytoonah University of Jordan, Jordan

Article Info

Article history:

Received Jan 11, 2019
Revised Apr 14, 2019
Accepted May 5, 2019

Keywords:

Non-functional requirements
Requirements classification
Security requirements
Software requirements
SVM

ABSTRACT

Software requirements with its functional and non-functional methods are the first important phase in producing a software system with free errors. The functional requirements are the visual actions that may easily evaluated from the developer and from the user, but non-functional requirements are not visual and need a lot of efforts to be evaluated. One of the main important non-functional requirements is security, which focuses on generating secure systems from strangers. Evaluating the security of the system in earlier steps will help to reduce the efforts of reveals critical system threats. Security threats found because of leaking of security scenarios in requirement phase. In this paper, we purpose an intelligent model to extract and evaluate security features from scenarios based on set of security system goals and a set of security requirements saved on rich story scenarios dataset (RSSD). This model will used a support vector machine (SVM) classifier to classify the security requirement based on RSS dataset. The using of SVM will enhance the overall process of evaluating the security requirements. The results show a significant enhancement in security improvements.

Corresponding Author:

Akram AbdelQader,
Department of Software Engineering,
Al-Zaytoonah University of Jordan,
Queen Alia airport road, Amman, Jordan.
Email: akrama@zuj.edu.jo

1. INTRODUCTION

The software requirements are the most important step in developing good systems. Software requirements illustrate and identify what must be delivered to the software users. System requirements are represented in a form of use cases using natural language [1]. Non-functional requirements represented as software behaviors that are implemented on software systems (such as authorizations and users authentications).

Achieving security requirements to test clues to insecure behaviors and then explore potential vulnerabilities, and most of these vulnerabilities arise from unexpected interactions between different system components [1]. In addition, the security of most component-based software development applications and systems is still not properly noticed [2]. In most software system, the problem of security is to predict and evaluate the security at the early stage of software development [2].

To insure the availability of security requirements in a given requirement scenarios, we proposed a new novel intelligent model to extract, classify and evaluate the security features from scenarios based on the generated RSSD that where the security requirements features are saved. Security features are evaluated from the scenarios by a set of acceptable system behaviors that show how these behaviors are shared among the system components. This paper consists of four sections: Section 2, previous literature. The methodology of the proposed work illustrated in Section 3, illustrates the proposed algorithms and the methodology of

collecting the non-functional security requirements. Section 4, presents the evaluation and the experimental results. Section 5, is the conclusion of this paper.

2. RELATED WORK

The non-functional requirements (NFR) elicitation is very important to produce working software, one of the main NFR is the security, the security requirements may elicit during the developing software lifecycle or after the development of software where we can detect the weak points and attacks that penetrate and patch. Besides that, many research studies found that eliciting security need high cost after producing software more than if we consider security in initial phases of producing software [3-6]. The weekness of security requirements in software system projects will introduce risk management system problems that need efficient computational techneques to find these security risk factors in the system [7, 8]. There are several studies that are made on testing scenarios and security testing. Scenario based specifications such as Message Sequence Chart (MSC) and its Labeled Transition System Analyzer (LTSA) which checks for system behavior [9]. Implied scenario detection for security testing reveals unexpected interactions between components [1]. Out of a formal specification language, a Use Case scenario is declared by extended UML 2.0 sequence diagrams to derive a test model to assist test designers with test-specific information for later execution [10]. Model-based testing (MBT) and test case selection techniques using triggers, guards, and genetic algorithm-based selection are used to detect system real faults [11].

In [12] the authors proposed a method to eliciting the nun-functional security requirements using the use cases. The proposed method was a systematic approach with emphasis on description and method guidelines. They extend the use case representation to include misuse. They also cover extra-functional requirements that imply security. Mustafa and Kamalrudin [13] proposed a new mathematical formulation to define the consistency validation rules of security requirements using best-practice template pattern library. The method was based on the security-related semi-formalized model, called SecEssential Use Case (SecEUC). The approach was realized with a proof of concept prototype. Shambhu and Mishra [2] proposed a suitable guiding principle based on requirement specification and analysis of the software architecture. The approach used by software engineers to develop secure component-based software products. Portugal et al. [14] they proposed a semi-automated process strategy for finding the possible NFRs in a text based on keywords by asking system stakeholders about list of qualities. The used catalogs based of the NFR Framework, as a supporting knowledge base. In [15] the authors proposed a dynamic model to collect and manage the requirements automatically. They proposed a compatible requirement template to the standard templates. The method was concerned on a functional requirements elicitation into a proposed template. Authors in [16] proposed a PROM model for predictive and optimization the risk in management with the perspective of risk requirements in software engineering. The work was based on practical scenario of software development practices in information technology. They use machine learning as an assessed for computationally cost effective analysis of risk factor based on different quality standards of software projects. In [17] they proposed classification and identification of NFRs in structured texts, they used an identification of keywords in texts for supervised methods to find security NFR, and they archived results with an average of 57% for F-measure.

3. THE PROPOSED METHODOLOGY

Functional and non-functional software requirements describe what the system can do and how it should work at the system level. System functional specifications include the description of system processes that is to say the interaction between the user and the system as well as between subsystems [10]. Collecting system requirements are based on the investigation to system processes. Descriptive software system requirements are the documented results of the intensive software analysis phase as they are collected through various techniques such as interviews, surveys, and sampling methods. Software security requirement focuses on producing secure systems to illuminate intruders' access control to the system. Security testing is motivated by addressing undocumented assumptions and areas of particular complexity to determine how a program can be broken [18]. Software security testing is heavily addressed during the software analysis phase to verify that it behaves as expected [19] using a set of security test scenario cases that are extracted from security test cases [1]. Security testing scenarios should identify vulnerabilities that arise from unexpected interactions between system components [1], and identifies the behaviors and interactions of system components [20].

In this paper, we propose a new method to generate a database contains security system features and security goals of non-functional security requirements. This database will be the dataset of rich story scenarios of non-functional security requirements. This dataset will be used in the requirement analysis phase

to classify and evaluate the security requirements from user scenarios. The proposed methodology and algorithms and are illustrated in Figure 1.

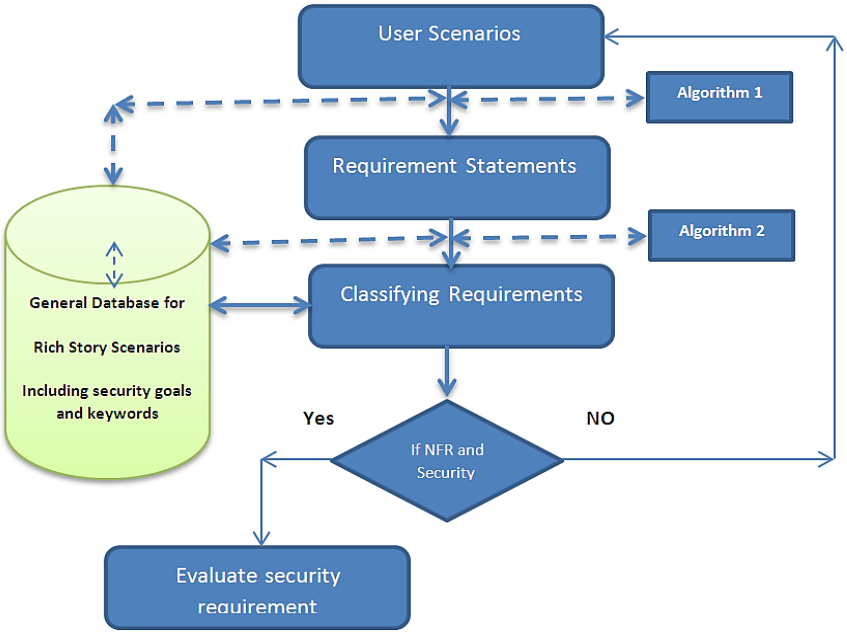

Figure 1. The the proposed methodology

3.1. Generating the System Security Goals and Security Requirements RSS Dataset

In this stage, we propose an algorithm to create and build the system security goals and security requirements RSS dataset. This dataset can be used in evaluating stage to find and evaluate the security requirements from the user scenario. In this section, we will generate a table of security goals and convert the user scenario into a set of requirement statements then these statements are converted into RSSD table. The system security goals are extracted based on expert users and developers to construct the system security goals and keywords table (SGKT). This table includes the system name, the security goals, security keywords and the required constraints in the system. Besides that, the set of requirement statements that extracted from the user scenario are saved in security requirement statements and security keywords table (SRSKT), this table includes the system name, the requirement statements, and the security keywords.

3.2. Generating the Security Goals, keywords and Requirements Dataset

In this algorithm we use the user scenario to generate the SSGT and SRST dataset. This dataset will be the trained dataset that used in the proposed intelligent model to find and evaluate the security requirements form user scenario in early stage using the proposed detection process. The algorithm steps illustrated as the following:

a) Create a security goals and keywords table (SGKT) that contains the system name, the security goals, and keywords as shown in Table 1. This table can be dynamically contained any number of systems with its attribute columns.
b) Create the security requirement statements and security keywords table (SRSKT); this table includes the system name (SNM), the requirement statements (RS), the security keywords (SK) and the requirement category type (RCT) as shown in Table 2. Each system name has many requirement statements each with relative security keywords and set of constraints.

Table 1. The Proposed System Security Goals Table (SGKT)

System Name (SNM)	Scenario Description (SD)	Security Goals Keywords (SGK)

Table 2. The Proposed Security Requirement Statements and Keywords Table (SRSKT)

System Names (SNM)	Requirement Statements (RS)	Security Keywords (SGK)	Requirement category (RCT)

We considered that these two tables are created by expert users and developer and the information inserted to these tables are tested and trained by expert software engineering developers and users. This dataset tables will be used as the base dataset in evaluation process as an early step to find the security requirements before the system started. Table 3; illustrate a sample of dataset example of security scenario and security goals, where the security goals specify what the system should prevent, not how it should accomplish that prevention. In addition, the requirement constraint must contribute to satisfaction of a security goal, therefore the security requirements are maintain as a constraint on a functional requirement and used as security keywords. Besides that, Table 4 shows an example of converting the scenarios into a set of requirement statements, security keywords, and requirement category

Table 3. Example of Security Scenario and Security Goals and Keywords

System Name (SNM)	Scenario Description (SD)	Security Goals Keywords (SGK)
System name1	The application shall protect the confidentiality of data. Also, it shall preserve the integrity of data, and shall promote the availability of data for authorized use. Besides that, it shall prevent/detect action on/to/with asset.	Protect, Confidentiality, Preserve, Integrity, Availability, Authorized, Prevent, Detect
System name2	The application shall require user identification and authentication by using a password that is at least 8 characters long and modify their passwords at least once a month. Also the application shall use a COTS public-key encryption and decryption package to ensure that confidential data remains secure. Furthermore, the application shall use the MD5 128-bit hash code to ensure the detection of corrupted messages.	Identification, Authentication, Encryption, Decryption, Confidential, Detection

Table 4. Example of Converting the Scenarios Into a set of Requirement Statements, Security Keywords, and Requirement Category

System Names (SNM)	Requirement Statements (RS)	Security Keywords (SGK)	Requirement category (RCT)
System name1	The application shall protect the confidentiality of data.	Protect, Confidentiality,	Non-functional Security requirement
	The application shall preserve the integrity of data	Preserve, Integrity	Non-functional Security requirement
	The application shall promote the availability of data for authorized use.	Availability, Authorized	Non-functional Security requirement
System name2	The application shall require user identification and authentication using a password that is at least 8 characters long and modify their passwords at least once a month.	Identification Authentication	Non-functional Security requirement
	The application shall use a COTS public-key encryption and decryption package to ensure that confidential data remains secure	Encryption Decryption Confidential	Non-functional Security requirement
	The application shall use the MD5 128-bit hash code to ensure the detection of corrupted messages.	Detection	Non-functional Security requirement

3.3. Convert the user Scenario Into a Set of Requirement Statement and Security Keywords

In this algorithm we convert the user scenario into set of requirement statements. These statements are scanned to extract the security goals and keywords based on the proposed dataset in section 3.2 using a support vector machine (SVM) classifier to detect the security keyword existence in the security goals, keywords and requirements dataset. The proposed algorithms are illustrated as below:

Algorithm 1: From user scenario into set of requirement statement and security keywords

1) Open SGKT and SRSKT (RSS) dataset tables.
2) Insert the system name and user scenario
3) Write the system name and user scenario into SGKT table.
4) Read the user scenario
   a) Correct the user scenario text for detection errors.
   b) Convert the user scenario text into set of statement lines.
   - Split the user scenario into lines as a requirement statements based on (.) dot punctuation mark remove stop characters by using the General Architecture for Text Engineering (GATE) [21].
   - Use standard tokenization, sentence splitting and stemming, as shown in classifier design in Figure 2.
5) For each requirement statement line, search for each word in the text using the detection method proposed in algorithm 2 by comparing it with the trained security keywords dataset.
   - If the word exist in security keywords dataset
   a. Accept the requirement statement as nun-functional security requirements
   b. Return the word as a security keyword
6) Repeat step 4 and 5 until all user scenarios finished.

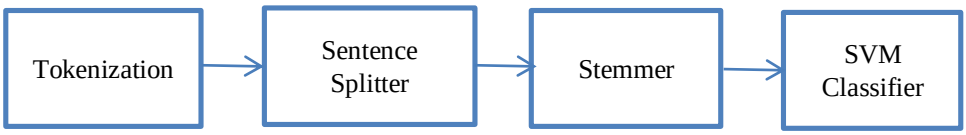

Figure 2. The classifier design

Algorithm 2: Classifying and Evaluating the Security features and Keywords using SVM Classifier

In this phase we use an intelligent method to detect and find the security requirements based on security keywords from the user scenario. This method use a detection technique based on SVM classifier to classify the non-functional requirements with the security goals and keywords. In this algorithm we use training database of saved security goals and keywords as the following:

1) Build machine learning classifier using the training database based on NFR classifier for requirements scenario statements.
2) Classify the input sentences into 2 major categories FR/NFR [22-24], focusing on NFR with 6 categories: security, efficiency, reliability, functionality, and usability/utility where defined as a quality standards in [24] and [25].
3) Classify the keywords of security class as (Protect, Confidentiality, Authorized, Authentication, Encryption, Decryption, Confidential and Detection)
4) Convert the resulted set of keywords into set of training/testing features. The set of used features are the unigram of the sentences' tokens, using its stem.
5) Use binary SVM classifier for each type of NFR, since some sentences contains two or more types of requirements. Figure 3 illustrates the evaluating the security keywords model using SVM classifier.

4. EXPERIMENTAL RESULTS

To evaluate the proposed methodology and algorithms we apply a set of experiments based our generated database. We use a Human Resource (HR) system documentations and scenarios that prepared from different legacy system based on a set of expert users and developers.

We use a sample of 55% from the available HR documentation in different categories in the training phase, and we use 45% of the HR documentations in the testing phase. This database is used to train the proposed model as in algorithm1 and algorithm2.

The resulted features are saved in the database to be used in testing phase. The results in Table 5 show a number of tested classes sentences per each user scenario documents, for set of NFR security features (PR: Protect, CO: Confidentiality, AZ: Authorized, AN: Authentication, EN: Encryption, DE: Decryption, DT: Detection).

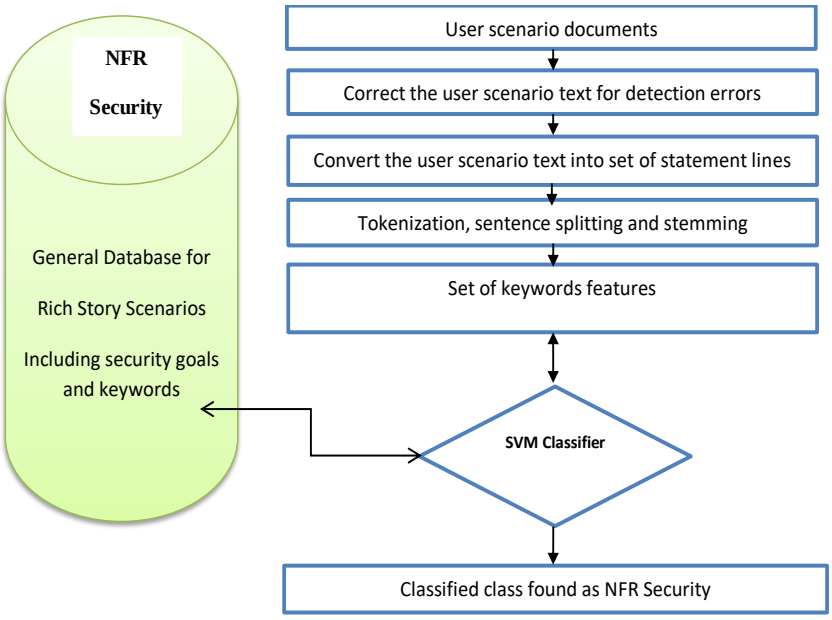

Figure 3. Classifying and evaluating the security keywords model using SVM classifier

Table 5. The Tested Classes for Each User Scenario Documents based on the set of NFR Security Features

HR System User Scenario documents	NR	FR	NFR	Security Classes						
				PR	CO	AZ	AN	EN	DE	DT
User scenario 1	14	31	22	1	1	6	7	2	2	3
User scenario 2	49	47	28	2	2	7	8	3	4	2
User scenario 3	65	98	45	5	4	11	13	5	6	1
User scenario 4	72	112	24	2	2	7	6	3	3	1
Total	200	288	119	10	9	31	34	13	15	7

4.1. Evaluation of Non Functional Requirements (NFR)

The SVM classifier is evaluated based on the tested requirements using the user scenarios from 4 different HR systems based on the proposed 7 security classes. The results evaluated using the metrics of Precision, Recall and F-Measure [21], defined as follows:

$$Precision = \frac{TP}{TP+FP},$$

$$Recall = \frac{TP}{TP+FN},$$

$$F\text{-}measure = \frac{2\ X\ Precision\ X\ Recall}{Precision+Recall}$$

where, TP (True Positive) is the number of correctly classified requirements, FP (False Positive) the number of requirements incorrectly classified, and FN (False Negative) the number of requirements incorrectly not classified. Results of the SVM classifier on the given 7 security classes are shown in Table 6.

Table 6. The Results of SVM Classifier on the 7 Security Classes

Security Classes	Precision	Recall	F-Measure
PR	90.9%	90.9%	90.9%
CO	90.0%	90.0%	90.0%
AZ	93.9%	91.2%	92.5%
AN	94.4%	94.4%	94.4%
EN	92.9%	92.9%	92.9%
DE	93.8%	93.8%	93.8%
DT	100.0%	87.5%	93.3%
Average	93.70%	91.53%	92.54%

5. DISCUSSION

The purposed model shows a significant improvement in identifying the non- functional security recruitments over unstructured text in the user scenario. This model was implemented and tested over different user scenarios. The proposed approach is fully automated method that can be used in eliciting requirements especially non-functional security requirements. Our results are compared with a semi-automated process strategy called NFRFinder that proposed in [14] the results was tested over structured text relative to requirements template and was not tested over unstructured text. The results of NFRFinder [14] was performed better in structured samples with a F1-measure of 72%. Besides that, our proposed model shows a high performance and a high accuracy rates with a F1-measure of 92.54%. Moreover, our proposed method can classify the security requirements into 7 classes based on our generated database. Table 7 illustrating the comparison between our proposed method and the methods proposed in [14] and [17]. The proposed method was tested for the 7 classes were trained in our database the method can be expandable by adding a new training dataset with new requirements. Also our method needs to be trained in different application systems to be compatible tool for any user scenario system in the future.

Table 7. The Comparison between our Proposed Method and the Methods Proposed in [14, 17]

	Precision	Recall	F-Measure
The proposed Method	93.70%	91.53%	92.54%
NFRFinder [14]	73.00%	61.00%	72.00%
Huang et al. [17]	56.7%	78.9%	57.00%

6. CONCLUSION

The proposed method will helps the developers and system analysts in classifying the requirements into FR and NFR. In addition, this model will classifying the NFR security requirements into 7 classes, these classes are trained to build a non-functional security requirements database. This database is used in testing the new requirements and classifying them into different security requirements features. The proposed model is tested using several HR requirements that are collected from different HR systems. The results show a high accuracy compared with manual methods in collecting the security requirements. In addition, the using of linear SVM classifier improves the accuracy and the performance of the process, where the security requirements classes are evaluated. The result shows high accuracy rates of 93.70%. This method is a new and novel approach to find and classifying the security requirements into set of security categories.

REFERENCES

[1] S. Al-Azzani and R. Bahsoon, "Using Implied Scenarios in Security Testing," *SESS'10, Cape Town, South Africa*, Copyright ACM, May 2010.

[2] Shambhu K. J. and R. K. Mishra, "Predicting and Accessing Security Features into Component-Based Software Development: A Critical Survey" *Software Engineering, Part of the Advances in Intelligent Systems and Computing book series (AISC)*, vol. 731, pp. 287-294.

[3] H. Mouratidis, et al., "Using security attack scenarios to analyze security during information systems design," *6th International Conference on Enterprise Information Systems*, 2004/ Available: roar.uel.ac.uk.

[4] G. McGraw, "Software Security: Building Security in," published by Addison Wesley Professional, Jan 2006.

[5] G. Hoglund and G. McGraw, "Exploiting Software: How to Break Code," Third Printing, Published By Addison-Wesley, Apr 2004.

[6] J. Viega, "The CLASP Application Security Process Volume 1.1 Training Manual," 2006. Available: http://www.owasp.org/index.php/OWASP_CLAS P_Project 7/2/2006.

[7] E. J. Jeong, et al., "Guidelines aimed at reducing the risks of user acceptance delay in the context of an IT service project management plan," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, pp. 832-839, 2015.

[8] S. Bhatia and J. Malhotra, "CSPCR: Cloud Security, Privacy and Compliance Readiness-A Trustworthy Framework," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, pp. 3756-3766, 2018.

[9] S. Uchitel, et al., "Detecting Implied Message Sequence Chart Specifications," *ESEC/FSE 2001, Vienna, Austria*, Copyright ACM, 2001.

[10] R. Loffler, et al., "Formal Scenario-based Requirements Specification and Test Case Generation in Healthcare Applications," *SEHC'10, May 3-4, 2010, cape Town, South Africa*, Copyright ACM, 2010.

[11] H. Hemmati, et al., "An Enhanced Test Case Selection Approach for Model-Based Testing: An Industrial Case Study," *FSE-18, Santa Fe, New Mexico, USA*, Copyright ACM, Nov 2010.

[12] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements engineering, Springer*, vol. 10, pp. 34–44, 2005.

[13] N. Mustafa and M. Kamalrudin, "A New Consistency Validation Approach to Enhance the Quality of Functional Security Requirements for Secure Software," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 10, pp. 73-76, 2018.
[14] R. L. Q. Portugal, et al., "NFRFinder: A Knowledge Based Strategy for Mining Non-Functional Requirements," *Proceeding SBES '18 Proceedings of the XXXII Brazilian Symposium on Software Engineering*, pp. 102-111, 2018.
[15] M. Lafi and A. A. AbdelQader, "A novel dynamic integrated model for automated requirements engineering process," *International Journal of Computer Applications in Technology*, vol. 56, pp. 292-300, 2017.
[16] S. Firdose and L. M. Rao, "PORM: Predictive Optimization of Risk Management to control Uncertainty Problems in Software Engineering," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, pp. 4735-4744, Dec 2018.
[17] J. C. Huang, et al., "Automated classification of non-functional requirements," *Springer in Requirements Engineering*, vol. 12, pp. 103-120, 2007.
[18] M. Howard and D. C. LeBlanc, "Writing Secure Code," Microsoft Press, Re-dond, WA, 2002.
[19] G. Wimmel and J. J¨urjens, "Specifiction-based test generation for security-critical systems using muttions," *in International Conference on Formal Engi-neering Methods (ICFEM)*, 2002.
[20] M. A. Babar and I. Gorton, "Comparson of scenario-based software architec-ture evaluation methods," *Asia Pacific Software Engineering Conference*, 2004.
[21] H. Cunningham, et al., "Text Processing with GATE," Gateway Press CA, University of Sheffield, Department of Computer Science, 2011.
[22] D. Firesmith, "Common Concepts Underlying Safety, Security, and Survivability Engineering," ser. Technical note. Carnegie Mellon University, Software Engineering Institute, 2003.
[23] D. Leffingwell and D. Widrig, "Managing software requirements: a unified approach," Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2003.
[24] M. Wolski, et al., "Software quality model for a research-driven organization-An experience report," *Journal of Software: Evolution and Process*, vol. 30, pp. e1911, 2018.
[25] "ISO/IEC 9126-1:2001 software engineering – product quality – part 1: Quality model."