SECRET: A New SECurity REquirements SpecificaTion Template

Hiba Hnaini, Raul Mazo, Paola Vallejo, Andres Lopez, Joël Champeau, Jose Galindo

SECRET: a new SECurity REquirements specificaTion template

Hiba Hnaini[1], Raúl Mazo[1] Paola Vallejo[2], Andres Lopez[2], Joël Champeau[1], and Jose Galindo[3]

[1] ENSTA Bretagne, Brest, France
hiba.hnaini@ensta-bretagne.org
[2] Universidad EAFIT, Medellín, Colombia
[3] Universidad de Sevilla, Sevilla, Spain

Abstract. The growing number of software and hardware system attacks has made security a critical factor in their design. To guarantee the safety of the system, security requirements must be established, but this is a difficult task since they are usually expressed in natural language. Researchers have proposed structured natural language templates for specifying security requirements to simplify this process. These templates must consider various security aspects, such as security criteria and mechanisms. This paper introduces the SECurity REquirements specificaTion (SECRET) template, developed through two action research cycles to address these issues. Initially, the SECRET template was created and tested by identifying and resolving problems in defining security requirements. Then, the SECRET template was applied to a use case, and it was concluded that all security requirements were correctly specified.

Keywords: Security requirement, Security requirements engineering, Natural Language, Template

1 Introduction

The past few years have seen a sharp rise in security threats to software and hardware systems, making security a critical factor to consider during the development process, beginning with the definition of requirements. However, creating accurate, simple, and non-complex security requirements is a time-consuming and laborious task. To address this issue, guides have been created to help security requirements engineers define their requirements.
Natural language (NL) is widely used for requirements elicitation and documentation due to its accessibility and minimal training requirements [3]. However, NL-based requirements often suffer from ambiguity, lack of clarity, inconsistency, incompleteness, vagueness, complexity, duplication, verbosity, implementation challenges, and untestability [12]. To address some of these problems, semi-structured natural language in templates has been proposed. Templates help structure requirements while retaining the advantages of NL, reducing early-stage defects in the development process. Adapting templates to the security domain requires consideration of specific security aspects, including security criteria and mechanisms. Previous work identified deficiencies in existing require-

ment templates [7], such as missing quantities, non-verifiable non-functional requirements, and insufficient specificity [14]. Other templates and libraries have been proposed to assist engineers in writing security requirements [9], [6]. However, these approaches had limited coverage of security aspects.
This paper introduces SECRET (SECurity REquirements specificaTion), a security requirements template building upon Mazo *et al.*'s template [14] while considering application and domain requirements, including those crucial for product lines [13] and self-adaptive systems [19] . SECRET provides structured guidance for specifying functional, non-functional, and security requirements. It seeks to answer two key research questions: *What should a well-structured security requirements template include?* and *How should security requirements engineers utilize this template?* The proposed template aims to (i) establish a guided process for specifying these requirements, (ii) distinguish problem and solution spaces, and (iii) differentiate domain from application requirements.
The rest of the paper is structured as follows: Section 2 explores templates and guidelines. Section 3 discusses the research method. Section 4 reports issues identified using the Mazo *et al.*'s template. Section 5 explains the SECRET template for security requirements specification. Section 6 presents the SECRET template evaluation. Finally, Section 7 covers the conclusions and future work.

2 Baseline and Related Work

Controlled natural language: Structured requirements often use controlled natural languages like Attempto Controlled English (ACE) [20], which enables clear domain-specific expressions. However, ACE does not cover security or self-adaptive systems. In contrast, Gellish, an application-independent language, facilitates system customization and data integration but lacks support for security criteria and mechanisms [17]. **Guidelines and templates:** Various works provide templates and guidelines for specifying requirements. EARS by Mavin *et al.* simplifies requirements into five templates but does not address security or domain-specific requirements, focusing on system requirements [12]. Mahmud *et al.* offer a structured ReSA language toolchain for more precise requirements but lacks coverage of security mechanisms and domain-specific needs [11]. Esser and Struss propose a template-based approach for functional testing but primarily focuses on test case generation from functional specs, with limited security criteria support [4]. Rupp *et al.* provide a requirements template emphasizing structure but may not suit all requirement types and lacks security and non-functional requirement details [18]. **Security requirements specification:** Several guides and templates aid in specifying security requirements. Kamalrudin *et al.* provide a comprehensive security requirements template covering authentication, integrity, confidentiality, availability, and non-repudiation [9]. However, it may not encompass all system security requirements and lacks domain-specific considerations. Firesmith promotes reusable templates for specifying security requirements at the asset level [6]. However, finding detailed information about these templates in various databases can be challenging, limiting their usability [1] [5] [2]. Souag *et al.* introduce AMAN-DA, a method offering domain and

application-level security requirements templates based on SIREN [21]. Pabuccu *et al.* present the Cube template for general software systems, adaptable for security and domain-specific requirements as needed [16]. **Self-adaptive systems and product lines requirements specification:** Mazo *et al.* [14] introduce a template for functional and non-functional requirements tailored to self-adaptive systems and product lines. It consists of nine sections covering conditions, system hierarchy, priority, activity characterization, objects, object details, conditionality, verification criteria, and RELAX statements. However, this template does not meet security requirements and lacks domain and application-level specifications. Proposed improvements include adding security criteria and mechanisms and improving the framework to simultaneously address domain and application requirements for better coverage, particularly for shared components.

3 Research method

Action research, chosen for this study [15, 23, 24], involves intervening to improve a situation and learn from it. This approach aims to resolve issues, offer practical solutions, and investigate phenomena in their natural context [10]. It is cost-effective and relies primarily on researchers, reducing validity threats. Action research follows five phases: diagnosis (identifying problems), action planning (identifying solutions), taking action (implementing a solution), evaluating (examining consequences), and specifying learning (interpreting results and initiating new cycles) [22]. In this work, we conducted two action research cycles, each representing security requirements specification for a system.

In the **first cycle**, we analyzed the security requirements of two systems: the Remote Patient Monitoring System [8] and an Electric Vehicle (EV) Charging System proposed by the European Network for Cybersecurity[4]. The Remote Patient Monitoring System is an IoT device for health data transmission, while the EV Charging System powers electric vehicles. We introduced the first version of the new semi-structured Natural Language template for specifying security requirements, considering three solutions: adapting prose-style requirements from [8] using templates by Mazo *et al.*, ACE, and Kamalrudin *et al.* In the **second cycle**, we assessed the requirements specification for Medical Domain Systems or Devices proposed by the ANSM (Agence nationale de sécurité du médicament et des produits de santé)[5], which are employed for collecting patients' medical data. During this cycle, we considered three solutions: prose-style security requirements, initial SECRET template, and enhanced SECRET template (version 2) described in this paper.

In each cycle, we followed these steps: ***(1) Diagnosing***: We encountered issues with security requirement specifications using prose style and Mazo *et al.*'s template initially. We conducted multiple mini-cycles to identify and resolve these problems, leading to the development of a new template. ***(2) Action Planning***: The Mazo *et al.* template could not adequately handle security aspects,

[4] Electric Vehicles Charging System
[5] ANSM'S GUIDELINE Cybersecurity of medical devices

such as criteria and mechanisms. We experimented with Kamalrudin *et al.*'s template and ACE in the first cycle, creating the SECRET template. In the second cycle, we refined SECRET and introduced Version 2, integrating elements from existing templates to maintain consistency. ***(3) Taking Action***: We identified security requirements gaps in each cycle and used the identified templates to establish a consistent pattern for requirement specification. ***(4) Evaluating***: We assessed if the templates covered 100% of security requirements, aiming for accurate representation while avoiding issues like ambiguity, complexity, or omission. ***(5) Specifying Learning***: After each cycle, we interpreted results to evaluate the strengths and limitations of the enhanced template. Collaboratively, we chose action research to assess the SECRET template's industrial effectiveness, conducting two cycles with different cases. However, we acknowledge that three cases may not fully validate the template. We used insights from the third case to improve it. Plans involve further refinement through multiple cycles with diverse industrial cases for real-world applicability.

4 Problems identified in the baseline templates

In the first cycle of our study, we addressed security requirements for the Remote Patient Monitoring System, evaluating three templates: Mazo *et al.*'s, ACE, and Kamalrudin *et al.*'s. This led to the initial SECRET template. Identified template shortcomings and gaps, including missing security criteria and mechanisms, are documented in two online sources[6][7]. Notably, the security criteria within requirements, like Integrity or Confidentiality, were often embedded in identifiers rather than explicitly stated. For instance, requirements like "Identification Requirement" or "Physical Protection Requirement" implied specific security criteria but lacked explicit mention. Additionally, security mechanisms were often omitted from requirements, although they were essential for guaranteeing security. For instance, an "Immunity Requirement" failed to specify the corresponding security mechanism, "Antivirus: Immunity." These issues provided insights for refining and enhancing the SECRET template.

In the second action research cycle, using SECRET Version 1, refined from the first cycle, we specified ANSM's medical devices security requirements. This cycle addressed three key issues: the separation between domain and application security requirements, the distinction between problem and solution spaces, and the inclusion of security norms or standards. The identified shortcomings during this cycle are thoroughly documented in an online source[8], complete with concise descriptions and examples for each issue. Firstly, there was no clear separation between domain-specific and application-specific security requirements. For instance, **R59** represented a domain-level requirement applicable to all connected medical devices, while **R40** pertained to specific devices within the same domain. Separating domain and application requirements would enhance clarity.

[6] Remote Patient Monitoring System Requirements
[7] Electric Vehicles (EV) Charging System Requirements
[8] Remote Medical Device Domain Requirements

Secondly, some security requirements, such as **R59** and **R40**, shared the same solution, applicable to various domain problems and potentially to application problems within the same domain. Separating the problem and solution spaces would facilitate reusing solutions for similar requirements. Lastly, the template lacked a dedicated section for specifying security standards or norms. For instance, **R25** mentioned complying with standard BS EN 50159. A standardized section for explicitly stating applicable norms or standards within the template would simplify its use for requirements engineers.

5 Proposed Template - SECRET

Through two action research cycles, we improved the Mazo *et al.*'s template [14], resulting in the first and second versions of the SECRET template. The first version, shown in Figure 1[9], was divided into 11 parts. Parts 1-4 corresponded to sections 1-4 of Mazo *et al.*'s template, part 5 to the security criteria, part 6 to the object part, parts 7 and 8 are the conditionality in object and additional object details conforming to the Mazo *el al.* template, part 9 corresponds to the security mechanism, and parts 10 and 11 to parts 8 and 9 of Mazo *et al.*'s template, respectively. The second version of the SECRET template, presented in Figure 2[10], was created to address three issues: (i) no separation of domain and application security requirements, (ii) no separation between problem and solution spaces, and (iii) missing capability to represent standards and norms in the template. This version was divided into three parts: domain specification in the problem space, application specification in the problem space, and domain and application specification in the solution space. Part 8 represents potential norms and standards associated with the requirements.

1-Conditions under which a behavior occurs. This component (1 - yellow) is identical in domain and application requirements. The options are: **(a) Requirements with logical conditions:** describe behaviors triggered only when a logical condition is achieved or when a sudden event happens ***IF*** *<condition or event>* ***THEN***. **(b) Requirements guided by the state:** describe behavior that must be completed while the system is in a specific state. ***WHILE*** | ***DURING*** *<activation state>*. **(c) Requirements with optional elements:** describe behavior that must be completed only if a specific feature is included. ***IN CASE*** *<included feature>* ***IS INCLUDED***. **(d) Requirements with temporary conditions:** describe a behavior that must happen after another behavior occurs. ***AFTER*** | ***BEFORE*** | ***AS SOON AS*** *<behavior>*. **(e) Requirements with complex conditions:** for complex conditional clauses, it is sometimes required to add keywords, such as When, While, or Where. It can help specify richer behaviors of the system.

[9] SECRET V1 : https://shorturl.at/bAVX5
[10] SECRET V2 : https://shorturl.at/antSV

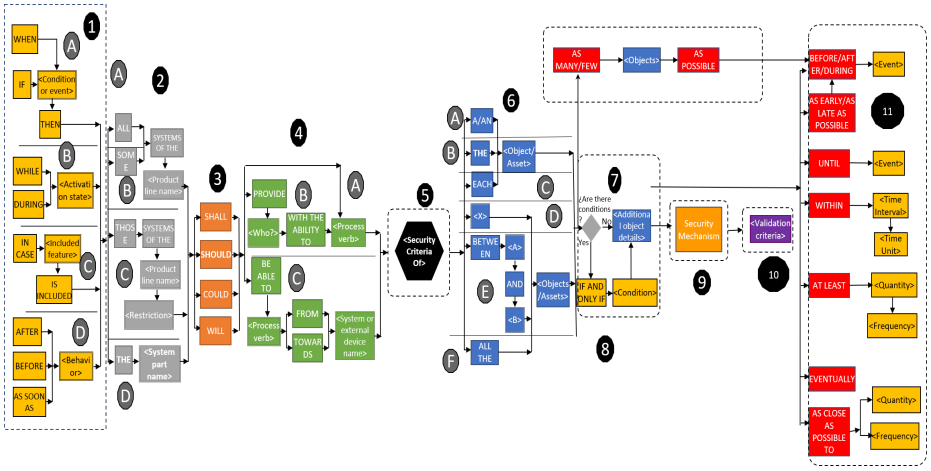

Fig. 1: SECRET Version 1

2-Family of systems, systems or parts of a system. This component is different in domain and application requirements. **System or part of system:** It is illustrated by part (2 - gray) of the application problem space in Figure 2. Since we are specifying a requirement for a specific known application, the following notation is used: Name of the system or a specific part of it. ***THE*** *<system or part name>* . **Family of systems:** It is illustrated by part (2 - gray) of the domain problem space in Figure 2. In this case, we specify requirements for a domain. Thus, the following notations are used: (a) Name of all or some systems of a product line. ***ALL SYSTEMS OF THE*** | ***SOME SYSTEMS OF THE*** *<product line name>* . (b) Name of the product line and condition or restriction of some product line systems. ***THOSE SYSTEMS OF THE*** *<product line name> <restriction>* .

3-The degree of priority. It is the component (3 - orange) of Figure 2. **(a) Essential requirements:** must be implemented to reach the success of the product or the product line. **SHALL** . **(b) Recommended requirements:** important but optional to reach the success of the product or the product line. **SHOULD** . **(c) Desirable requirements:** desirable, but not necessary, used to improve the user experience and customer satisfaction. **COULD** . We also added the fourth priority will. It indicates optional requirements where it is up to the technical team to implement them or not **WILL** .

4-The activity It is the component (4 - green) of Figure 2. There are three types of activities: **(a) Autonomous activity:** no user is involved, which indicates that the (sub) system or systems start and complete the behavior autonomously. *<process verb>* . **(b) User interaction:** the (sub) system or systems provide a user with the ability to employ a specific behavior started or completed by a user (actor) that interacts with the system(s). ***PROVIDE*** *<who?>*

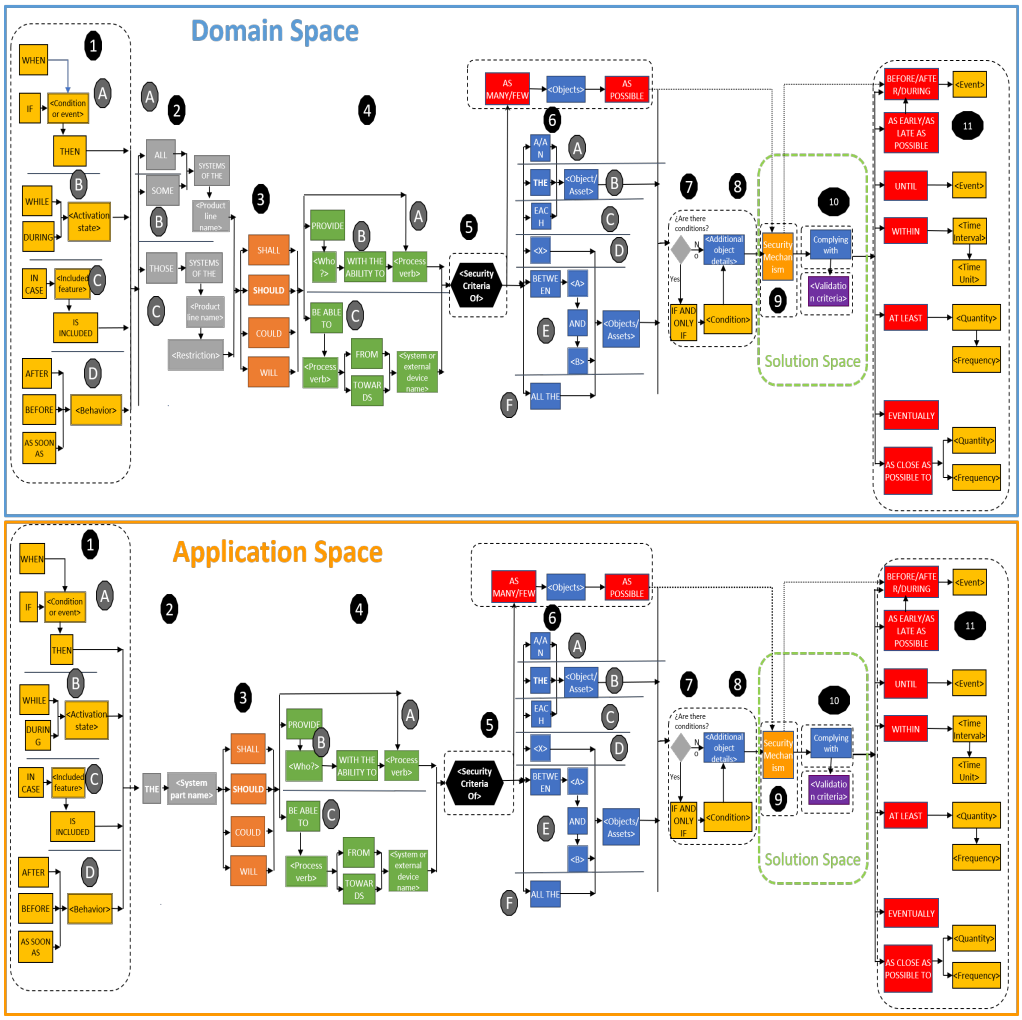

Fig. 2: SECRET Version 2

WITH THE ABILITY TO *<process verb>*. **(c) Interface requirement:** the system executes a behavior that relies on another entity. ***BE ABLE TO*** *<process verb>* **(i) If the external system executes the behavior:** ***FROM*** *<system or external device name>* **(ii) If the behavior is executed by the system and interacts with another system or external device:** ***TOWARDS*** *<system or external device name>*.

5-Security Criteria This component (5 - black) of Figure 2 represents the security goal or goals to be achieved by the requirement. It has the following notation: *<Security Criteria>*. An example of a security criterion is Authentication. <The Remote Monitoring System>$_{system}$<SHALL>$_{priority}$<guarantee

>$_{processVerb}$<**the authentication**>$_{securityCriteria}$<of person or device>$_{object}$< with whom it is interacting>$_{additionalObjectDetails}$<by identity verification >$_{securityMechanism}$.

6-Object or Objects The object(s) or asset(s) are represented by the available ranges: **(a) Single object.** ***ONE*** *<object>* . **(b) A specific object.** ***THE*** *<object>* . **(c) Each object of a set.** ***EACH*** *<Object>* . **(d) Multiple objects.** *<X> <objects>* , where X is the number of objects. **(e) Range of objects.** ***BETWEEN*** *<A>* ***AND*** *<B> <objects>* . **(f) All objects in a set.** ***ALL THE*** *<objects>* An "of" is added before the object component as the requirement represents the need to guarantee the security criterion of an object, for example, the integrity of user data. <During transit>$_{Condition}$<The Remote Monitoring System>$_{system}$<SHALL>$_{priority}$<guarantee>$_{processVerb}$<the integrity>$_{securityCriteria}$<**of medical data**>$_{object}$<from an intruder >$_{additionalObjectDetails}$.

7-Conditionality in the object It is represented by part (9 - yellow) of Figure 2. It has the following notation: ***IF AND ONLY IF*** *<condition>* .

8-The complementary details It is represented by component (10 - blue) of Figure 2. It has the following notation: *<additional object details>* .

9-Security Mechanism The security mechanism of the requirement is considered a part of the solution since it indicates how and what mechanism to use to achieve the security criteria. It is represented by part (7 - orange) of Figure 2. It has the following notation: *<Security Mechanism>* . An example of a security mechanism for authentication is identity verification. <The Remote Monitoring System>$_{system}$<SHALL>$_{priority}$< guarantee>$_{processVerb}$<the authentication>$_{securityCriteria}$<of person or device>$_{object}$<with whom it is interacting>$_{additionalObjectDetails}$<**by identity verification**>$_{securityMechanism}$.

10-Validation Criterion or Standard The validation criteria of a security requirement can be a security norm or standard that it has to comply with. It is represented by part (8 - purple) in Figure 2. It has the following notation: ***COMPLYING WITH*** *<Validation Criteria>* .

11-Relax requirements statements for self-adaptive systems It is represented by component (11 - red) of Figure 2. The options are: (a) Should maximize or minimize the occurrence of a specific number of objects, as many or as few as possible. ***AS MANY*** | ***AS FEW*** *<object>* ***AS POSSIBLE*** . (b) Must be achieved before, during, or after a specific event. ***BEFORE*** | ***AFTER*** | ***DURING*** *<event>* . (c) Describes something that should be achieved as early as possible or delayed as late as possible. ***AS EARLY AS POSSIBLE*** |

AS LATE AS POSSIBLE. (d) Must be sustained until an upcoming event. ***UNTIL*** *<event>*. (e) Must be sustained during a specific time interval. ***WITHIN*** *<time interval> <time unit>*. (f) Should reach a minimum frequency or time (until infinity). ***AT LEAST*** *<quantity> <frequency>*. (g) Behavior that should happen eventually. ***EVENTUALLY***. (h) Specifies something that happens repeatedly but has flexible frequency or quantity. ***AS CLOSE AS POSSIBLE TO*** *<quantity> <frequency>*.

6 Preliminary Evaluation

In evaluating the SECRET template, we organized two groups for each cycle: business analysts and technical reviewers. During the first cycle, one group used Mazo *et al.*'s template, while the other used SECRET - V1 to define security requirements for the Remote Patient Monitoring System and Electric Vehicles (EV) Charging System. Subsequently, technical reviewers assessed the resulting requirements. The Remote Patient Monitoring System had 15 prose-style requirements, while the EV Charging System comprised 101 requirements (excluding vendor-specific ones). When the first group's business analysts redefined the security requirements using Mazo *et al.*'s template, they encountered uncertainties. The technical reviewer identified issues in 17 Remote Patient Monitoring system requirements and all 101 requirements for the EV Charging System when using Mazo *et al.*'s template. These issues are categorized in Table 1. Conversely, in the other group, business analysts successfully re-specified the systems' requirements using the SECRET template, with all 17 and 96 security requirements receiving approval without any noted issues from the technical reviewer.

Table 1: Problems identified in the requirements in the first cycle

Problem identified	Nb. of requirements using the Mazo *et al.*'s template	Nb. of requirements using SECRET
Remote Patient Monitoring System		
Missing Security Criteria	17	0
Missing Security Mechanism	3	0
Electric Vehicles (EV) Charging System		
Missing Security Criteria	101	0
Missing Security Mechanism	101	5

In the second cycle, we identified security requirements for medical domain devices, resulting in 64 prose-style requirements. After excluding seven unrelated requirements, the remaining 57 prose-style requirements were transformed into 61 security requirements using the SECRET template. However, compatibility issues arose with the first version of the SECRET template, which was extended in the second version. These issues are outlined in Table 2.

In evaluating security requirement templates, the first cycle revealed critical issues using Mazo et al.'s template, highlighting gaps in security coverage for complex systems. The second cycle demonstrated significant improvements in

the SECRET template, attributing its success to iterative development, expert input, and meticulous alignment with prose-style requirements. Feedback from the initial cycle informed targeted refinements, ensuring enhanced clarity, comprehensive coverage, and seamless integration. Rigorous testing further solidified the second version's effectiveness, eliminating issues and showcasing the template's evolution toward robust and reliable security specifications.

The final SECRET template has been integrated into the VariaMos tool[11] for simplified domain and application security requirement specification. VariaMos offers two languages, "Application Requirements AC"[12] and "Domain Requirements AC"[13] implementing the template. These languages facilitate the execution, creation, editing, and deletion of security requirements and establish traceability relationships. This integration enhances the accessibility and usability of the SECRET template for security requirement specification.

Table 2: Problems identified in the requirements in the second cycle

Problem identified	Nb. of requirements using SECRET Version 1	Nb. of requirements using SECRET Version 2
No separation between domain and application security	61	0
No separation between problem and solution spaces	61	0
Missing standards or norms	4	0

7 Conclusions and Future Work

In this paper, we introduced the SECRET template for specifying security requirements. We built upon Mazo *et al.*'s template, aiming to address its limitations in capturing security requirements. Our research, conducted through action research, focused on enhancing the representation of security requirements. We identified that Mazo *et al.*'s template could be improved by incorporating structural and security components from the ACE and Kamalrudin *et al.* templates. Our study included three industrial cases, revealing significant enhancements in security requirements concerning cost, time, and quality standards. To further strengthen our findings, future work involves experimenting with additional reference templates beyond Mazo *et al.*, ACE, and Kamalrudin *et al.* and comparing their outcomes with the SECRET template. Extending the evaluation to diverse industrial cases and domains will provide a broader perspective on the SECRET template's utility. Additionally, we plan to integrate a security ontology into the SECRET template to generate security requirements based on specific security criteria, enhancing security coverage. The SECRET template and its associated ontology form essential guides within a comprehensive framework for specifying, formalizing, and analyzing hardware and software system security.

[11] http://www.variamos.com/
[12] https://shorturl.at/cPSV1
[13] https://shorturl.at/loqF1

References

1. Engineering security requirements. The Journal of Object Technology 2 (2003)
2. Specifying good requirements. Journal of Object Technology 2, 77–87 (2003)
3. Denger, C., Berry, D.M., Kamsties, E.: Higher quality requirements specifications through natural language patterns. Proceedings 2003 Symposium on Security and Privacy pp. 80–90 (2003)
4. Esser, M., Struss, P.: Obtaining models for test generation from natural-language-like functional specifications. Proc. of 18th International Workshop on Principles of Diagnosis (2007)
5. Firesmith, D.: Generating complete, unambiguous, and verifiable requirements from stories, scenarios, and use cases. Journal of Object Technology 3(10), 27–39 (2004), http://www.jot.fm/contents/issue_2004_11/column3.html
6. Firesmith, D.: Specifying reusable security requirements. J. Object Technol. 3(1), 61–75 (2004)
7. Hnaini, H., Mazo, R., Vallejo, P., Galindo, J., Champeau, J.: Taxonomy of Requirements Specification Templates. In: SoftEng 23. Venice, Italy (2023), https://hal.science/hal-04105054
8. Jaiswal, S., Gupta, D.S.: Security requirements for internet of things (iot). Advances in intelligent systems and computing 508, 419–427 (2017)
9. Kamalrudin, M., Mustafa, N., Sidek, S.: A template for writing security requirements. In: Asia Pacific Requirements Engeneering Conference. pp. 73–86. Springer (2017)
10. Koshy, E., Koshy, V., Waterman, H.: Action Research in Healthcare. SAGE Publications (2010), https://books.google.fr/books?id=Vb1w8mKAbScC
11. Mahmud, N., Seceleanu, C., Ljungkrantz, O.: Resa tool: Structured requirements specification and sat-based consistency-checking. In: 2016 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 1737–1746 (2016)
12. Mavin, A., Wilkinson, P., Harwood, A., Novak, M.: Easy approach to requirements syntax (ears). In: Proceedings of Requirements Engineering Conference, 2009. RE '09. 17th IEEE International. pp. 317–322. IEEE, United States (2009)
13. Mazo, R.: Guía para la adopción industrial de líneas de productos de software. Universidad Eafit (2018), https://books.google.fr/books?id=d7cjvwEACAAJ
14. Mazo, R., Jaramillo, C.M.Z., Vallejo, P., Medina, J.M.: Towards a new template for the specification of requirements in semi-structured natural language. J. Softw. Eng. Res. Dev. 8, 3 (2020)
15. O'Brien, R.P.: An overview of the methodological approach of action research (2008)
16. Pabuccu, Y.U., Yel, I., Helvacioglu, A.B., Asa, B.N.: The requirement cube: A requirement template for business, user, and functional requirements with 5w1h approach. International Journal of Information System Modeling and Design (IJISMD) 13(1), 1–18 (2022)
17. van Renssen, A.: Gellish: an information representation language, knowledge base and ontology. ESSDERC 2003. Proceedings of the 33rd European Solid-State Device Research - ESSDERC '03 (IEEE Cat. No. 03EX704) pp. 215–228 (2003)
18. Rupp, C., Simon, M., Hocker, F.: Requirements engineering und management. HMD Praxis der Wirtschaftsinformatik 46, 94–103 (2014)
19. Sawyer, P., Mazo, R., Diaz, D., Salinesi, C., Hughes, D.: Constraint programming as a means to manage configurations in self-adaptive systems. IEEE Computer pp. 1–1 (2012)

20. Schwitter, R., Fuchs, N.: Attempto controlled english (ace) a seemingly informal bridgehead in formal territory (1996)
21. Souag, A., Mazo, R., Salinesi, C., Comyn-Wattiau, I.: Using the aman-da method to generate security requirements: a case study in the maritime domain. Requirements Engineering 23(4), 557–580 (2018)
22. Susman, G.I.: Action research: a sociotechnical systems perspective. Beyond method: Strategies for social research 95(113), 95 (1983)
23. Susman, G.I., Evered, R.D.: An assessment of the scientific merits of action research. Administrative Science Quarterly 23, 582–603 (1978)
24. Wieringa, R., Moralı, A.: Technical action research as a validation method in information systems design science. In: Peffers, K., Rothenberger, M., Kuechler, B. (eds.) Design Science Research in Information Systems. Advances in Theory and Practice. pp. 220–238. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)