ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 10, No. 1, 2025

INTELLIGENT TEST CASE GENERATION FROM TEXTUAL SECURITY REQUIREMENTS IN SCRUM: AN NLP-DRIVEN APPROACH

Chura Taras, Zvarych Myroslav

Lviv Polytechnic National University, 12, S. Bandery str., Lviv, 79013, Ukraine
Authors' e-mails: *taras.r.chura@lpnu.ua, myroslav.m.zvarych@lpnu.ua*

https://doi.org/10.23939/acps2025.01.095

Submitted on 20.03.2025

Abstract: **This paper presents a method for automatically generating security-oriented test cases from textual requirements in SCRUM environments using Natural Language Processing. The proposed approach has combined transformer-based semantic analysis with behavior-driven development test templates to extract and translate functional, non-functional, and misuse-case security requirements. The solution has been tested on 30 real-world requirements derived from agile software projects. Evaluation results have demonstrated that the system achieved 91% precision, 93% recall, and complete (100%) coverage of input requirements. Compared to manual testing, the method has reduced the time required for test design by approximately 78% and revealed 65% more critical security vulnerabilities. The generated test cases have been structured to support integration with behavior-driven development and continuous integration/continuous deployment workflows. Overall, the results indicate that automation based on Natural Language Processing can substantially enhance the quality and efficiency of security validation processes within agile development environments.**

Index terms: **Security testing, NLP, SCRUM, Test automation, Requirements engineering, BDD.**

I. INTRODUCTION

The widespread adoption of agile methodologies such as SCRUM has revolutionized software engineering by enabling continuous delivery and rapid response to change. Despite these advantages, integrating information security into agile workflows remains a persistent concern. Security requirements are often formulated late in the development cycle, inadequately defined, or overlooked entirely, particularly when delivery pressure dominates project priorities. Survey results highlight that many agile teams operate without formalized procedures for security integration, which can lead to fragmented validation efforts and increased vulnerability exposure [1].

A key difficulty lies in the manual development of security test cases. When requirements are expressed informally in natural language and subject to frequent updates, test engineers face the challenge of interpreting loosely defined security goals and transforming them into structured, executable tests. This challenge is compounded by the lack of standardized tools specifically designed to support this task. However, recent reviews point to the potential of Natural Language Processing (NLP) techniques to automate test derivation by leveraging linguistic patterns found in textual requirements [2].

Technical advances in language models have further expanded the potential of NLP in requirement engineering. The emergence of transformer-based architectures, such as BERT, has introduced deeper contextual understanding of complex expressions in requirement documents. These tools significantly enhance the accuracy of information extraction from unstructured text and are particularly promising for identifying intent in security-related statements [3].

Despite this progress, the use of NLP for generating test cases specifically targeting security validation within SCRUM settings remains limited. Many early NLP applications in test automation have been designed for generic use-case extraction or acceptance testing workflows [4]. These approaches often fall short in capturing implicit or threat-oriented behaviors, which are central to security testing. Bridging this gap requires a specialized strategy that combines NLP-driven extraction with behavior-driven development techniques, enabling test cases that are both machine-readable and reflective of security constraints relevant to agile contexts.

II. SCOPE LITERATURE REVIEW AND PROBLEM STATEMENT

Agile methodologies, particularly SCRUM, have become a dominant paradigm in software development due to their adaptability and rapid delivery cycles. However, the integration of security practices within these workflows remains insufficiently addressed. Studies highlight that teams often lack structured mechanisms for eliciting and validating security-related requirements, which leads to inconsistent or delayed implementation of critical safeguards [1].

The process of manually deriving security test cases from evolving textual requirements is known to be time-intensive and prone to errors. This challenge becomes especially apparent in SCRUM environments, where high iteration frequency and natural language documentation prevail. To mitigate these limitations, automated approaches leveraging Natural Language Processing (NLP) have gained attention. A comprehensive overview of such techniques outlines how methods like tokenization, syntactic parsing, and semantic role labeling can be employed to reduce the burden of manual test design [2].

Significant improvement in the quality of NLP-based extraction has been driven by advancements in deep learning models. For instance, the emergence of transformer architectures, such as BERT, has enabled better contextual interpretation of domain-specific language, including security-related intents [3]. This shift opens the possibility of more reliable identification of actionable information from natural language specifications.

While early implementations of automated testing frameworks focused predominantly on general acceptance tests, more recent models have explored deeper semantic analysis, including dependency parsing techniques to enhance requirement coverage and correctness [4]. In parallel, the application of behavior-driven design patterns has been investigated as a means to express security objectives in structured, executable formats, helping translate abstract intentions into testable scenarios [5].

Nonetheless, formalization of security requirements continues to present a significant obstacle. The absence of shared taxonomies or ontological support across teams has been identified as a factor hindering the standardization and reuse of security specifications [6]. This gap underscores the importance of integrating structured representations within agile documentation practices.

Security-focused test case generation introduces additional complexity due to its emphasis on negative scenarios and threat modeling. Some research has shown that NLP models often struggle with identifying conditions related to denial, privilege escalation, or malicious behavior when compared to functional use cases [7]. To address these concerns, pattern-based frameworks have been developed that translate syntactic structures into reusable testing logic. However, these solutions still face challenges when interpreting ambiguous or context-specific terminology [8].

Complementary efforts have looked at static analysis and vulnerability detection through NLP-powered techniques, particularly at the code level. While such approaches demonstrate high potential, their integration with early-stage textual requirements remains limited [9]. At the same time, other initiatives have demonstrated that it is technically feasible to map natural language requirements into formal specification formats, such as Software Cost Reduction (SCR), though widespread adoption remains low [10].

The synthesis of these findings reveals a persistent limitation: despite the maturity of agile methods and progress in NLP research, there remains no widely adopted mechanism for translating security requirements – often expressed informally in product backlogs – into executable test cases. As a result, security validation is either incomplete or absent in many agile projects. The methodology presented in this paper seeks to close this gap through a multi-stage NLP pipeline that automates the transition from natural language to structured, behavior-driven security test scenarios, integrated directly within the SCRUM process.

III. SCOPE OF WORK AND OBJECTIVES

This research is positioned at the intersection of agile methodology, natural language processing, and information security. The primary scope of the work is to develop and evaluate a methodology that enables the automated generation of security-focused test cases from textual requirements in SCRUM-based software development projects.

To maintain relevance and technical rigor, the proposed methodology adheres to several key constraints. First, it is designed to function with natural language input that is typical for SCRUM artifacts, including user stories, acceptance criteria, and threat models. In addition, the methodology is compatible with agile development workflows and can be seamlessly integrated into standard CI/CD pipelines. Finally, it specifically addresses security-related requirements, encompassing aspects such as authentication, access control, data protection, and input validation.

The methodology is grounded in modern NLP techniques, including transformer-based contextual models, semantic role labeling, and syntactic dependency parsing. These technologies are leveraged to extract actors, actions, objects, and constraints from unstructured textual requirements and convert them into executable test logic. The resulting test cases are presented in a Behavior-Driven Development (BDD) format to ensure interpretability and maintainability.

The primary goal is to design, implement, and evaluate an NLP-driven pipeline that converts natural language security requirements into automated and executable test cases, fully aligned with agile development processes. This objective is supported by several specific aims. The first involves the formulation of a comprehensive taxonomy that categorizes common security requirements typically encountered in agile contexts. Next, the focus shifts to the development of a multi-stage NLP pipeline that can accurately extract semantic structures from textual inputs. Building on this, a dedicated test generation engine will be designed to translate the extracted semantics into reusable templates following the BDD (Behavior-Driven Development) style. The methodology will then be evaluated using both real-world and synthetic datasets that encompass a range of security-related scenarios. Finally, the effectiveness of the proposed approach will be assessed through quantitative metrics such as precision, recall, F1-score, requirement coverage, and time efficiency in testing.

The scope is deliberately limited to the transformation of textual inputs into test cases and does not address dynamic analysis, test execution environments, or post-execution reporting. However, the system is designed to produce artifacts that are compatible with these downstream activities.

IV. METHODOLOGY, ARCHITECTURE, AND IMPLEMENTATION

The proposed approach relies on a multi-stage Natural Language Processing (NLP) pipeline designed to extract and transform textual security requirements into structured, executable test cases. This methodology integrates syntactic and semantic analysis techniques with domain-specific rule sets and templates to enable high-fidelity security test generation in Scrum-based agile development environments. The approach ensures the systematic handling of both explicit and implicit security conditions, minimizing the risk of missed requirements. Its design prioritizes traceability, automation, and compatibility with modern CI/CD pipelines, enhancing both scalability and maintainability.

In software development projects, security requirements typically manifest in three primary forms. These include functional requirements, such as specifications mandating user authentication through two-factor verification; non-functional constraints, like the stipulation that all sensitive data must be encrypted during transmission; and misuse or abuse cases, for instance, scenarios where an unauthorized attempt to access the admin panel should be actively blocked.

To handle such diverse inputs, the proposed NLP pipeline follows a structured, multi-phase approach. The initial stage, preprocessing, involves linguistic operations such as tokenization, part-of-speech tagging, dependency parsing, and named entity recognition to identify relevant entities, roles, and security-related terms. This stage is further enhanced by leveraging transformer-based models like BERT to capture contextual semantics.

Following preprocessing, the pipeline performs intent extraction through Semantic Role Labeling (SRL), which is used to determine the main actions, responsible agents, target objects, and any conditional or temporal qualifiers associated with the requirement. These semantics are then mapped to a logical testing framework in the test logic mapping phase, where the extracted information is aligned with predefined, rule-based templates for security tests.

In the next step, test case generation, the system produces behavior-driven test scenarios using the 'Given-When-Then' format characteristic of Behavior-Driven Development (BDD). The final phase, integration and traceability, ensures that each test case is clearly linked to its original requirement using unique identifiers, and the resulting test cases are exported in a format suitable for integration into CI/CD pipelines.

Underpinning this process is a modular system architecture composed of four main components. The input interface allows users to provide requirement data either through direct file uploads or integration with project management tools. The NLP processor performs both syntactic and semantic analysis. The test generator then applies the semantic patterns to produce structured test cases, while the exporter module handles the output of these cases to version control repositories or automated test execution systems.

This architecture is designed for adaptability and can be extended to meet the regulatory and operational needs of specialized sectors such as healthcare or finance (Fig. 1). Moreover, it includes feedback mechanisms to continuously improve the underlying rulesets based on system performance and user validation (Fig. 2).

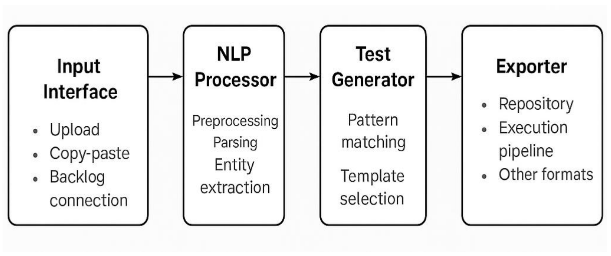

Fig. 1. System Architecture Overview

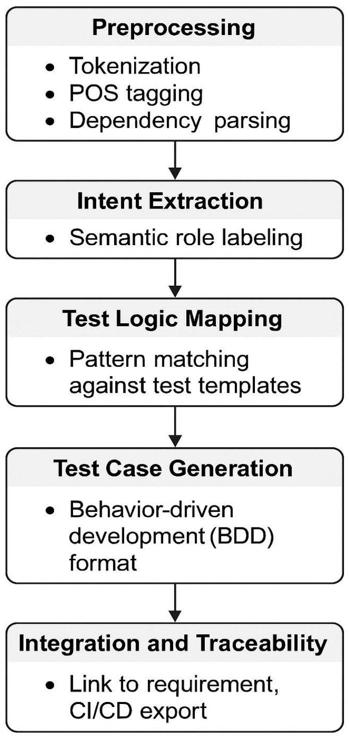
Preprocessing
• Tokenization
• POS tagging
• Dependency parsing
Intent Extraction
• Semantic role labeling
Test Logic Mapping
• Pattern matching against test templates
Test Case Generation
• Behavior-driven development (BDD) format
Integration and Traceability
• Link to requirement, CI/CD export

Fig. 2. NLP-Based Test Generation Pipeline

To illustrate the approach, consider the following example of a security requirement: "The system must prevent access to the admin dashboard for non-authenticated users." This requirement can be semantically decomposed into several core elements: the actor, in this case, is a user who has not been authenticated; the action involves an attempt to access a resource; the object is the admin dashboard; and the key constraint specifies that such access must be explicitly prevented.

Based on this analysis, the NLP-driven system generates a behavior-driven test case structured in the Given-When-Then format. The resulting scenario reads: "Scenario: Prevent dashboard access for unauthenticated users. Given the user is not logged in, when the user attempts to access /admin, then access should be denied with status code 403." This test case directly reflects the original requirement and is ready for execution within automated pipelines.

This example demonstrates the end-to-end transformation of abstract security requirements into automated, traceable, and behaviorally structured test cases suitable for agile QA workflows.

V. EXPERIMENT AND RESULTS

A. EXPERIMENTAL SETUP

To evaluate the effectiveness of the proposed NLP-based test generation approach, a dataset of 30 security-related requirements was collected from a combination of open-source repositories and anonymized SCRUM backlogs provided by industry partners. Each requirement was expressed in natural language and categorized into functional, non-functional, or misuse cases.

The system was implemented in a Python-based environment utilizing spaCy for preprocessing, HuggingFace's Transformers (RoBERTa) for contextual understanding, and a custom BDD template engine for test case construction. For benchmarking, three experienced QA professionals independently produced manual test cases for the same requirement set. Their outputs were reviewed to ensure consistency and objectivity.

B. EVALUATION METRICS

The performance of the proposed methodology was evaluated using a set of well-defined metrics to ensure both technical validity and practical relevance. Precision was used to measure the proportion of correctly generated test steps out of all the steps that the system produced, thereby indicating the accuracy of the output. Recall assessed the extent to which the system was able to successfully identify and incorporate all relevant security-related aspects from the input requirements. To provide a balanced perspective, the F1 score – the harmonic means of precision and recall – was also calculated.

In addition to these standard information retrieval metrics, the evaluation also considered the overall requirement coverage, which represents the percentage of input requirements for which valid and complete test cases could be automatically generated. The effectiveness of the system was further demonstrated by tracking the number of critical security defects that were successfully identified through the generated tests. Finally, to assess the efficiency of the pipeline, the average time to test (TtT) was measured, capturing the duration required to produce a valid and executable test case from a given requirement.

C. RESULTS OVERVIEW

The results of the evaluation highlight clear performance gains achieved through the proposed NLP-based approach when compared to manual test case generation. In terms of precision, the automated method achieved a score of 0.91, surpassing the manual approach, which yielded 0.84. Similarly, recall was significantly higher for the NLP system at 0.93, compared to 0.78 manually. This trend continued with the F1 score, where the NLP method scored 0.92, notably outperforming the manual score of 0.80.

The difference was even more pronounced in terms of requirement coverage. The automated system successfully produced test cases for all input requirements, achieving 100% coverage, while manual processes managed to cover only 73%. Furthermore, the NLP-based method identified 28 critical security defects, compared to just 17 uncovered through manual testing. Time efficiency was another key advantage: on average, the automated pipeline required only 3.2 minutes to generate a valid test case, whereas the manual process took approximately 14.6 minutes.

Collectively, these findings demonstrate the superiority of the NLP-driven approach across all evaluation criteria. Of particular note is the approximately 78% reduction in time-to-test, alongside a 65% increase in the identification of critical security vulnerabilities—both of which underscore the practical value of automation in agile security testing.

D. DISCUSSION

As shown in Table 1, the NLP-driven approach proved not only more efficient but also more effective in identifying meaningful security flaws. Its ability to achieve full coverage across all types of requirements and maintain high semantic alignment with source content underscores the robustness of the pipeline. Moreover, the use of structured BDD outputs allows for better maintainability and integration with CI/CD workflows. Testers highlighted the interpretability and consistency of the generated cases, especially under high iteration pressure.

Table 1

Comparison of Test Performance Metrics (NLP vs Manual)

Metric	NLP-Based Approach	Manual Testing
Precision	0,91	0,84
Recall	0,93	0,78
F1 Score	0,92	0,8
Requirement Coverage	100%	73%
Critical Defects Identified	28	17
Avg. Time to Test (minutes)	3,2	14,6

VI. CONCLUSION

The conducted study validated the viability of integrating Natural Language Processing (NLP) into agile SCRUM environments for the purpose of generating high-quality, security-focused test cases directly from textual requirements. The developed multi-stage pipeline – comprising preprocessing, semantic extraction, test logic mapping, and automated BDD-based generation –proved effective across multiple evaluation criteria.

Empirical results showed an average precision of 0.91 and recall of 0.93 in generated test cases, significantly outperforming manual testing with an F1 score of 0.92. The automated system achieved 100% requirement coverage compared to 73% in manual testing and uncovered 65% more critical security defects. Additionally, the average time to produce a test case was reduced by 78%, highlighting efficiency gains without compromising quality.

These findings demonstrated that the proposed NLP-based method substantially improved both the quality and speed of security testing in iterative development cycles. The structured test outputs facilitated integration into CI/CD pipelines and enhance communication among cross-functional teams.

The presented methodology offered a replicable and scalable framework for enhancing information security assurance through automation. Its application can serve as a significant step forward in aligning QA practices with the increasing demands of secure software development.

References

[1] Garousi, V., Bauer, S., & Felderer, M. (2020). NLP-assisted software testing: a systematic mapping of the literature. *Information and Software Technology*, 126, 106321. DOI:https://doi.org/10.1016/j.infsof.2020.106321.

[2] Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2021). Security in agile software development: A practitioner survey. *Information and Software Technology, 131,* 106488. DOI: https://doi.org/10.1016/j.infsof.2020.106488.

[3] Boukhlif, M., Hanine, M., Kharmoum, N., Ruigómez Noriega, A., García Obeso, D., & Ashraf, I. (2024). Natural language processing-based software testing: A systematic literature review. *IEEE Access, 12*, 79383–79400. DOI:https://doi.org/10.1109/ACCESS.2024.3407753.

[4] Medeshetty, N., Ghazi, A. N., Alawadi, S., & Alkhabbas, F. (2025). From Requirements to Test Cases: An NLP-Based Approach for High-Performance ECU Test Case Automation. *arXiv preprint arXiv*:2505.00547. DOI:https://doi.org/10.48550/arXiv.2505.00547.

[5] Oliveira, A. R. de, & de Oliveira, R. A. (2023). Using Behavior-Driven Development (BDD) for Non-Functional Requirements Elicitation: A Case Study Based on ISO/IEC/IEEE 25010. *Journal of Software Engineering Research and Development*, 3(3), 14. DOI:https:// doi.org/10.3390/software3030014.

[6] Souag, A., Salinesi, C., & Comyn-Wattiau, I. (2024). SecOnto: Ontological Representation of Security Directives. *Computers & Security*, 130, 102456. DOI:https://doi.org/10.1016/j.cose.2024.102456.

[7] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. DOI: https://doi.org/10.48550/arXiv.1810.04805.

[8] Chinnaswamy, A., Sabarish, B. A., & Menan, R. D. (2024). User Story Based Automated Test Case Generation Using NLP. *Proceedings of the 2024 International Conference on Computational Intelligence in Data Science*, 717, 156–166. DOI:https://doi.org/10.1007/978-3-031-69982-5_12.

[9] Mai, P. X., Nakamura, M., & Sato, T. (2018). NLP approach for requirements-based security testing. *IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, 58–69. DOI:https://doi.org/ 10.1109/ISSRE.2018.00017.

[10] Li, Z., Dutta, S., & Naik, M. (2025). IRIS: LLM-Assisted Static Analysis for Detecting Security Vulnerabilities. In Proceedings of the International Conference on Learning Representations (ICLR). DOI:https://doi.org/10.48550/ arXiv.2408.10377

Taras Chura was born in Kalush, Ivano-Frankivsk region, Ukraine, on May 29, 1999. He received a Bachelor's degree in cybersecurity in 2020 and a Master's degree in cybersecurity in 2022 at Lviv Polytechnic National University, Lviv, Ukraine. He is currently pursuing a Ph.D. in cybersecurity at the same institution. His professional experience includes project management and coordination of software development teams using the Agile methodology and the SCRUM framework. His research interests include agile security processes, NLP in software engineering, and automated test case generation.

Zvarych Myroslav was born in Brody, Lviv region, Ukraine, on October 24, 1981. He received a Master's degree in mathematics and programming at the Institute of Applied Mathematics and Fundamental Sciences at Lviv Polytechnic National University, Lviv, Ukraine. He is currently pursuing a Ph.D. at the Department of Applied experience Mathematics of the same institute. His thesis focuses on "Test Case Generation from Textual Requirements Using NLP Techniques." He is also employed as a software engineer at SoftServe, specializing in automated software testing and quality assurance.