ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

UDC 004.42:004.415.2:004.052.42 **doi: 10.32620/reks.2025.4.16**

Oleg ODARUSHCHENKO[1,2], Olena ODARUSHCHENKO[1], Oleksii STRIUK[2], Viacheslav SHAMANSKIY[2], Petro HROZA[3]

[1] ***Poltava State Agrarian University, Poltava, Ukraine***
[2] ***LLC RadICS, Kropyvnytskyi, Ukraine***
[3] ***Military Institute of Telecommunications and Information Technologies named after the Heroes of Kruty, Kyiv, Ukraine***

AUTOMATING REQUIREMENTS TRACEABILITY IN PROJECT DOCUMENTATION USING TRACETREND TOOL: MODEL, DESIGN AND APPLICATION

*The **object** of the study is a formalized model of requirements traceability in project documentation for hardware-software systems. **The subject matter of the research** encompasses the application of mathematical modeling and tool-based approaches to automate the traceability process, focusing on the design and functionality of the TraceTrend software tool. **The primary goal** of the study is to improve the quality and integrity of requirements management by implementing traceability mechanisms that ensure logical consistency, hierarchical correctness, and complete test coverage across all project documentation stages. **The research tasks include**: identifying challenges related to manual requirements tracing in safety-critical domains; constructing a formal mathematical model based on set theory, binary relations, and directed graphs; defining binary matrices for requirement inheritance and test coverage; developing automated analysis techniques for traceability conditions; integrating the model into the TraceTrend tool; and demonstrating its applicability through a real-world case study. **The study employed the following methods**: mathematical modeling of binary relations, model-based testing, static analysis of documentation structures, and the use of Boolean matrix operations for verifying coverage and consistency. **As a result of the research**, a formal model of requirements traceability was created and implemented in the TraceTrend tool. The tool enables automated extraction of requirement identifiers, construction of traceability matrices, and verification of coverage and logical completeness. The application of TraceTrend has shown its effectiveness in identifying undocumented requirements, broken dependencies, and gaps in test coverage early in the project lifecycle. **Conclusions.** The integration of formal models and traceability tools significantly strengthens the reliability and auditability of requirements management processes in engineering projects. TraceTrend has proven to be a valuable instrument for improving documentation quality and supporting compliance with standards such as IEC 61508 and ISO/IEC/IEEE 29148. Although the tool requires initial configuration for requirement markup, its benefits in enhancing visibility, consistency, and verification readiness justify its adoption in high-assurance development environments. The study confirms the necessity of embedding formal traceability analysis into standard project workflows to ensure both structural rigor and regulatory compliance.*

Keywords: *traceability analysis; mathematical modeling; formal verification; TraceTrend tool; test coverage; documentation consistency; safety standards; automated analysis; system engineering.*

1. Introduction

1.1. Motivation

In modern project activities, especially in the development of hardware and software systems for critical or technically complex applications, significant attention is devoted to requirements management throughout all stages of the project life cycle. The quality of requirement formulation, organization, and interrelation affects not only the functionality of the final product but also its compliance with safety, reliability, and quality standards [1, 2].

One of the key aspects of requirements management is requirements traceability–the process of establishing, maintaining, and analyzing links between various types of requirements (e.g., system, functional, non-functional, and interface), as well as between requirements and test scenarios, implementation components, risks, defects, and other project artifacts. Traceability is a prerequisite for validation, verification, change management, and audits. Within project documentation, it ensures logical consistency and completeness.

In typical hardware-software or system projects, there is a need to efficiently handle large volumes of tex-

tual documentation (requirements, specifications, technical assignments, detailed design descriptions, test plans, reports), where requirements often have a complex hierarchical structure and the links between them are not explicitly expressed. This complicates manual maintenance of traceability, especially in the context of frequent changes, collaborative development, and multi-level review processes. To support automated analysis of project documentation and enable formal traceability of requirements, this study proposes a mathematical model based on the concepts of set theory, binary relations, directed graphs, and Boolean matrices. Based on this model, the authors present the TraceTrend tool, designed to automatically build traceability links through configurable requirements markup in project documents. TraceTrend enables users to identify requirements, establish hierarchical relationships, verify test coverage, and detect traceability violations in accordance with international standards.

This paper provides:

– a formal description of the mathematical model of requirements traceability;

– a demonstration example based on real project documentation;

– an analysis of the TraceTrend as a tool to implement the traceability process;

– a list of possible improvements to the tool considering project management requirements.

The obtained results can be used to automate documentation checks, support requirements validation, and integrate traceability into the overall project management system.The research lies within the domain of information technologies, specifically in the areas of software engineering, requirements engineering, software development lifecycle (SDLC) management, and the design of hardware-software systems for critical applications. The work also intersects with systems engineering, functional safety, and tool-assisted quality assurance of project documentation. In the design of complex IT systems–particularly those that are safety-critical, security-related, or high-reliability–requirements traceability plays a key role. It involves establishing connections between requirements, implementation components, test cases, and risk factors. In practice, requirements traceability can be performed manually without using specialized software tools, or with the support of dedicated traceability tools. Existing traceability solutions range from simple scripts for identifying and linking requirements in textual documents to comprehensive and sophisticated industrial platforms integrated into Application Lifecycle Management (ALM) systems [3]. However, in small to medium and even relatively large-scale projects, or within research settings, there is a demand for a flexible, configurable tool that enables traceability within textual documentation. In this context, the development of mathematical models that provide formal traceability validation and instrumented analysis of documentation links becomes highly relevant.

The problem is formulated as the need to provide a mathematically justified and tool-supported process for requirements traceability that:

– detects inconsistencies or missing links in documentation;

– allows the construction of a requirement hierarchy and control of test coverage;

– integrates into project documentation without requiring a complex engineering environment.

This is particularly important for producing verifiable, understandable, reproducible, and certifiable documentation–especially in projects related to safety and quality.

Requirements traceability has been studied in the context of requirements engineering [4, 5], requirements management [6], and traceability modeling [7], as well as within commercial tools such as IBM DOORS, Polarion, and Reqtify [8]. Despite the availability of proprietary solutions, the creation of an open and adaptive traceability tool with formal validation of dependencies in project documentation remains an open challenge.

Unresolved aspects of the problem include:

– developing a mathematical model that accounts for not only direct and reverse dependencies but also for violations of traceability;

– designing a flexible configuration model for traceability that is independent of any specific ALM platform;

– ensuring full verification of requirements based on marked-up textual documents.

The aim of the study is to develop a mathematically grounded and tool-supported approach to requirements traceability in project documentation, considering structural hierarchies, test coverage, and functional safety standards.

The main objectives of the study are as follows:

1. To construct a mathematical model of requirements traceability based on sets, binary relations, and directed graphs.

2. To develop a demonstrative application of the model to real-world multi-level project documentation.

3. To implement traceability using the TraceTrend tool, with consideration for configurable rule sets.

4. To analyze violations of traceability conditions according to functional safety standards.

5. To propose directions for improving the TraceTrend tool in order to expand its capabilities for practical project applications.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

1.2. State of the art

In modern software engineering, requirements traceability is one of the central topics–both in the context of development quality assurance and in ensuring compliance with software lifecycle standards. Traceability enables artifact tracking across all stages of project activity: from requirements analysis to verification, release, and maintenance [9]. It is a key component for demonstrating that all safety-related requirements have been properly implemented and tested. Similarly, the ISO/IEC/IEEE 29148 [2] standard on requirements engineering emphasizes the importance of traceability for ensuring the clarity, completeness, and consistency of requirements–where consistency refers to the property of a requirement set that ensures the absence of contradictions between individual requirements or their components.

From a scientific perspective, requirements traceability is an active research area in software engineering. Scientific studies explore various aspects of traceability, including models for representing links between requirements, methods for automatically establishing and maintaining these links (e.g., based on natural language processing or information retrieval), and evaluation of the effectiveness of different traceability approaches [10, 11].

Over the past decades, the scientific community has proposed a variety of traceability methods, such as:

– graph-based traceability models that formalize links between artifacts [12];

– semantic analysis-based traceability [13];

– machine learning approaches for automatic extraction of traceability links from textual sources [14, 15].

Despite significant progress, several challenges remain unresolved, including:

– implementing simple traceability in textual documentation without requiring complex environments;

– validating traceability using formal rules;

– aligning tests with requirements without manual link specification.

Currently, a wide range of software tools exists to support the requirements traceability process. These tools range from simple spreadsheet-based solutions to comprehensive Requirements Management (RM) systems.

The main functional categories of traceability tools include:

– enterprise-grade ALM and requirements management platforms, such as BM DOORS / DOORS Next Generation**,** Polarion ALM**,** and Jama Connect offer extensive capabilities for creating, managing, tracing, and analyzing requirements throughout the development lifecycle. These tools typically support multiple types of traceability links, visualization of traceability matrices and dependency graphs, and integration with other development tools [16]. However, their disadvantages include high licensing costs, complexity of deployment, and a steep learning curve, which limits their applicability in small-scale or academic projects.;

– test management-oriented tools, such as Jira with the Xray plugin, TestRail, HP ALM, and Confluence, provide traceability from requirements to test cases and test results. These tools support requirements coverage analysis and tracking of validation status, particularly in agile development environments [17]. At the same time, their functionality is primarily focused on testing, and integration with broader requirements engineering processes is limited, making it difficult to achieve complete lifecycle traceability;

– model-based design and UML-oriented tools, such as Enterprise Architect or Rational Rhapsody, can support traceability from requirements to model elements, ensuring alignment between abstract models and specified requirements. Similar approaches to linking requirements with architectural and design artifacts are discussed in recent studies on embedded and safety-critical systems [18, 19]. Nevertheless, these tools usually lack automated verification of test coverage, require advanced modeling expertise, and provide limited integration with textual requirement specifications;

– lightweight and manual approaches**,** commonly used in smaller projects, where requirements traceability is maintained using spreadsheets such as Microsoft Excel or Google Sheets. Although this approach is less automated, it may be sufficient for simple use cases and is frequently recommended in practitioner-oriented guidelines [20]. However, spreadsheet-based traceability is error-prone, difficult to maintain at scale, and does not provide formal validation mechanisms [21]. This contrast between enterprise-grade ALM platforms and simplistic spreadsheet-based approaches illustrates a gap in current practice. On one hand, professional ALM systems offer formal rigor but are costly, complex, and require substantial training. On the other hand, spreadsheets are accessible and low-cost but error-prone and lack any formal validation of traceability. This gap has arisen because resource-constrained projects and academic environments often cannot afford heavy ALM platforms, yet they require stronger guarantees than spreadsheets can provide. Recent research efforts propose automated and semi-automated traceability approaches based on structured features, neural networks, or model-based techniques [20 - 22], but these solutions are typically tightly coupled to specific toolchains or development paradigms.

TraceTrend is designed precisely to bridge this gap by combining mathematical rigor with the accessibility and simplicity of lightweight, document-oriented tools. However, none of the existing solutions fully satisfies the

needs of projects that rely on document-centric workflows, operate under resource constraints, or require mathematically validated traceability without deploying enterprise-level platforms. This gap motivated the development of TraceTrend as a lightweight, configurable, and standalone tool that integrates formal modeling with practical document analysis.

1.3. Objectives and Approach

The objective of this article is to develop and validate an instrumental approach to requirements traceability in project documentation by means of the TraceTrend tool. The proposed solution aims to automate the identification of requirements, construction of their hierarchical relationships, verification of test coverage, and detection of inconsistencies in accordance with international standards of functional safety and documentation quality.

To achieve this objective, the following research approach has been adopted. First, a formal mathematical model of requirements traceability was constructed, based on set theory, binary relations, directed graphs, and Boolean matrices. This model provides a rigorous framework for representing inheritance between requirements, their association with documents, and their coverage by test cases. Second, the model was integrated into the TraceTrend tool, which performs automatic document scanning, requirement recognition through configurable regular-expression markup, and construction of traceability matrices.

The approach enables automated analysis of documentation consistency and completeness without reliance on large-scale ALM systems. By verifying traceability conditions and detecting coverage gaps, the TraceTrend tool ensures structural correctness and enhances compliance with safety and quality standards such as IEC 61508 and ISO/IEC/IEEE 29148. In contrast to existing solutions that are either manual and error-prone or embedded in complex industrial platforms, this study proposes a lightweight and flexible tool-supported method that can be effectively applied in resource-constrained projects, research environments, and educational contexts.

Structure of the article is the following.

Section 1. Introduction. Motivation for requirements traceability and its role in engineering projects is presented, along with challenges of manual tracing and limitations of existing industrial tools.

Section 2. Mathematical Model of Requirements Traceability. Theoretical foundations based on sets, binary relations, directed graphs, and Boolean matrices are developed. Formal definitions of inheritance, coverage, and test result functions are provided.

Section 3. Case Study: TraceTrend Tool. The implementation of the proposed model in the TraceTrend tool is described. The tool's algorithm, functional capabilities, and application examples are detailed.

Subsection 3.1. Application Algorithm. Step-by-step workflow of TraceTrend for document markup, analysis, matrix construction, coverage verification, and reporting.

Subsection 3.2. Advantages of Using TraceTrend. Benefits such as error reduction, improved consistency, and applicability in small-to-medium projects are highlighted.

Section 4. Discussion. The significance of automated traceability analysis is evaluated, and directions for improving TraceTrend (support of new formats, integration with ALM tools) are outlined.

Section 5. Conclusions. Summary of contributions, including formal model development, tool implementation, and validation results, with emphasis on compliance and practical applicability.

2. Mathematical Model of Requirements Traceability

Let us define the following finite sets:

- $D = \{d_1, d_2, ..., d_n\}$ – the set of documents that contain requirements;
- $RID = \{r_1, r_2, ..., r_k\}$ – the set of unique requirement identifiers;
- $TID = \{t_1, t_2, ..., t_m\}$ – the set of test cases identifiers.

Within the framework of the mathematical model for requirements traceability, the following set is introduced:

$$TR = \{Pass, Fail, N/T\}, \quad (1)$$

where: TR – is a finite set of test execution outcomes that reflects the verification status of each requirement in accordance with the established traceability links.

The elements of the set TR are defined as follows:

- Pass – the test case has been executed and the requirement has successfully passed verification;
- Fail – the test case has been executed but the result is unsatisfactory, the requirement has not passed verification;
- N/T (Not Tested) – the requirement is not yet covered by any test case or the corresponding test has not been executed. This status serves as an indicator of incomplete test coverage.

Furthermore, we introduce a relation denoting the association between a requirement and the document in which it is formally specified:

$$\delta \subseteq D \times RID, \quad (2)$$

where δ is a binary relation that determines in which document a specific requirement is formally recorded.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Although δ is formally defined as a binary relation, it represents an association (incidence) relation between requirements and documents, rather than a structural or dependency relation. Its purpose is to identify the document in which a requirement is formally recorded.

The inheritance relation among requirements is defined as a directed binary relation over the set of requirements, inducing an acyclic or partially ordered structure. This relation may be represented using a directed graph or a dependency tree:

$$\tau \subseteq \mathrm{RID} \times \mathrm{RID}. \quad (3)$$

In contrast, the relation τ represents a structural dependency between requirements and is therefore interpreted as a directed relation that can be represented by a graph or a dependency tree.

The proposed model does not introduce an explicit binary relation between documents. Instead, chains of traceability between documents are formed indirectly through hierarchical relationships between requirements and their association with specific documents. If a high-level requirement (e.g., a system requirement) is associated with a system requirements document, and its derived lower-level requirements are associated with software requirements, module specifications, V&V plans, and V&V reports, the sequence of these associations constitutes a traceability chain across documents. Consequently, document chains such as "System Requirements → Software Requirements → Module Requirements → V&V Plans → V&V Reports" are represented in the model as a composition of the requirement inheritance relation and the association relation between requirements and documents.

The coverage relation is a binary relation indicating the traceability between requirements and test cases. A test case t_j covers requirement r_i, if $(r_i, t_j) \in \theta$:

$$\theta \subseteq \mathrm{RID} \times \mathrm{TID}. \quad (4)$$

The test result function $\rho: \mathrm{TID} \rightarrow \mathrm{TR}$, is a mapping from the set of test case identifiers (TID) to the set of test outcomes (TR), such that each test case is assigned exactly one result value.

The model can be interpreted as a directed graph:

$$G = (V, E). \quad (5)$$

where $V = \mathrm{RID} \cup \mathrm{TID}$, $E = \tau \cup \theta$. This enables the visualization of traceability as a tree or directed acyclic graph, where nodes represent requirements and tests, and edges represent inheritance and coverage relations.

To represent the traceability structure as a single graph, requirements and test cases are modeled as distinct types of vertices within a unified vertex set. Although requirement identifiers and test case identifiers represent different artifact types, they are treated uniformly as graph nodes for the purpose of traceability analysis.

To formally represent hierarchical and traceability links, binary relation matrices are used. These matrices allow:

- compact representation of relationships between elements;
- automated validation of completeness and trace consistency;
- application of linear algebra methods for structural analysis.

The requirement inheritance matrix M_τ - is a binary matrix of size k×k, where k = |RID| - is the number of requirements in the project. The formal representation of the requirement inheritance matrix is as follows:

$$M_\tau \in \{0,1\}^{k \times k}, \quad (6)$$

where

$$M_\tau[i,j] = \begin{cases} 1, \text{ if } (r_i, r_j) \in \tau \\ \quad (\text{reqiurement } r_j \text{ is references to } r_i); \\ 0, \text{ otherwise;} \end{cases}$$

A simple example of a successor matrix is given below.

Let's consider a set of 6 requirements:

$$R = \{r_1, r_2, r_3, r_4, r_5, r_6\}.$$

Let the parent-child relationships between requirements be defined as: $r_1 \rightarrow r_2$, $r_1 \rightarrow r_3$, $r_2 \rightarrow r_4$, $r_3 \rightarrow r_5$, $r_5 \rightarrow r_6$.

Then the successor matrix $M_\tau \in \{0,1\}^{6 \times 6}$ is:

$$M_\tau = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The traceability matrix M_θ - is a two-dimensional binary matrix of size k×m, where k=|RID| - is the number of requirements in the project, and m =|TID| - is the number of test cases. The formal representation of the requirement traceability matrix is as follows:

$$M_\theta \in \{0,1\}^{k \times m}, \quad (7)$$

where

$$M_\theta[i,j] = \begin{cases} 1, & \text{if } (r_i, t_j) \in \theta_\theta \\ & \text{(test case } t_j \text{ tests the requirement } r_i); \\ 0, & \text{otherwise;} \end{cases}$$

When there exists an inheritance relation $r_1 \rightarrow r_2$, coverage achieved for r_2 should also be partially propagated to r_1. For example, if Test_02 verifies requirement r_2, then r_1 is indirectly covered through its dependency on r_2. This can be formally represented using the transitive closure of the inheritance and coverage matrices:

$$M_\theta^* = M_\tau \times M_\theta, \quad (8)$$

where M_θ^* – is the extended coverage matrix that incorporates indirect coverage induced by inheritance relations.

Thus, if $M_\tau[r_1, r_2] = 1$ (meaning r_1 depends on r_2) and $M_\theta[r_2, \text{Test\_02}] = 1$, then in the extended coverage matrix $M_\theta^*[r_1, \text{Test\_02}] = 1$.

This approach ensures that requirement coverage reflects not only direct test links but also indirect satisfaction through dependent requirements.

A simple example of a traceability matrix:

$$M_\theta = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}.$$

In this example, requirement r_5 (column 5 of the matrix) has no associated tests, as indicated by zeros in both columns of the traceability matrix. This means that r_5 is a documented requirement but remains uncovered, i.e., not validated by any test case. Such situations represent potential risks, since requirements without test coverage may lead to undetected defects or non-compliance with project standards. One of the key advantages of the TraceTrend tool is its ability to automatically highlight these uncovered requirements, ensuring that no specification item is omitted during verification.

Test Results Matrix

(TR - Pass/Fail/N/T) – 2 tests × 6

requirements has the following form:

$$M_T = \begin{bmatrix} P & F & N/T & N/T & N/T & P \\ N/T & P & P & P & N/T & N/T \end{bmatrix}.$$

This matrix combines traceability with actual test results: for example, r_2 is tested by two test cases, but one of them failed (F). This immediately indicates the risk of the requirement not being satisfied.

A simplified example for an extended coverage matrix is as follows:

$$M_\theta^* = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix},$$

where row 1 = Test_01, row 2 = Test_02; column 1 = r_1, column 2 = r_2.

As a result, after considering inheritance, requirement r1 additionally 'inherits' coverage by test Test_02.

The theoretical framework presented in Section 2, including the formal definitions of relations, matrices, and evaluation metrics, provides the foundation for the development of the TraceTrend software tool. The next section introduces the architecture and practical implementation of this tool, demonstrating how the proposed concepts are applied in real project documentation

3. Case Study

TraceTrend is a software tool developed to automate the process of requirements traceability within project documentation. Its primary objective is to minimize the time required for traceability execution and to reduce the likelihood of errors that may arise when modifying requirements or their interconnections. The tool performs automatic detection and validation of requirements in documents based on regular expressions and constructs traceability matrices that reflect the relationships between requirements and other project elements, such as tests or design decisions.

The core functional capabilities of TraceTrend include:

– scanning documents to detect requirements and their relationships;

– automatic generation of traceability matrices based on the identified links;

– test coverage verification and requirement status tracking;

– impact analysis of changes in requirements on the overall project structure.

The components of the mathematical model described above – namely, the set of documents (D) containing requirements, the set of requirement identifiers (RID), the set of test cases (TID), and the set of test results (T) – constitute the foundation for building traceability matrices. These matrices allow tracking the relationships between requirements, test cases, and other documentation elements. Based on this mathematical model, the TraceTrend tool performs automatic scanning of project documents to identify and validate requirements, and to generate traceability matrices. The resulting matrix represents the relations between requirements and test cases, enabling the assessment of whether all

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

requirements have been verified through testing, and whether any omissions or errors exist in the documentation.

3.1. TraceTrend Tool Application Algorithm

The TraceTrend tool operates according to the following algorithm:

1. Document Markup. The user defines regular expression templates to identify requirements and their relationships within project documents. These expressions can be configured according to project-specific conventions, enabling TraceTrend to be used with various types of documentation – from technical requirements to test reports.

2. Document Analysis. Once the requirements and tests contained in the document are marked with the appropriate tags, TraceTrend proceeds with its analysis. During this process, the tool detects requirements, identifies relationships among them, and determines the coverage of requirements by corresponding test cases. The working environment of TraceTrend is shown in Fig. 1.

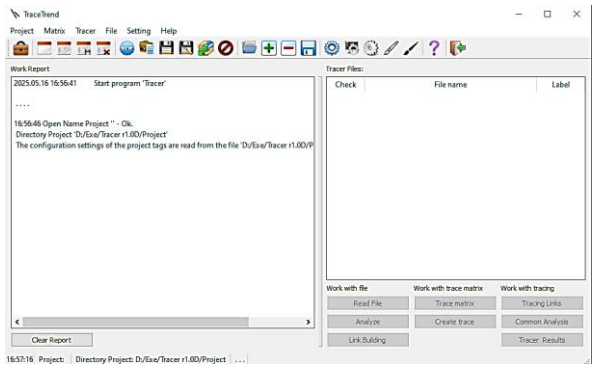

Fig. 1. Working Environment TraceTrend

3. Traceability Matrix Construction. Based on the identified links between requirements and test cases, TraceTrend constructs a traceability matrix. This matrix clearly shows which requirements are associated with corresponding test cases, whether additional tests are needed, and whether any violations exist in the relationships between requirements. An example of the constructed traceability matrix is shown in Fig. 2.

4. Test Coverage Verification. Each test case is checked for compliance with the corresponding requirements. If a requirement is not covered by any test, this is indicated in the results, helping to identify gaps in the testing process. Test coverage verification is shown in Fig. 3.

5. Reporting and Result Analysis. The tool generates reports on identified errors, missing requirements, and inconsistencies within the documentation.

This allows project managers to promptly detect issues at any stage of project execution and apply the necessary corrective actions. The summary table of traceability results is presented in Fig. 4.

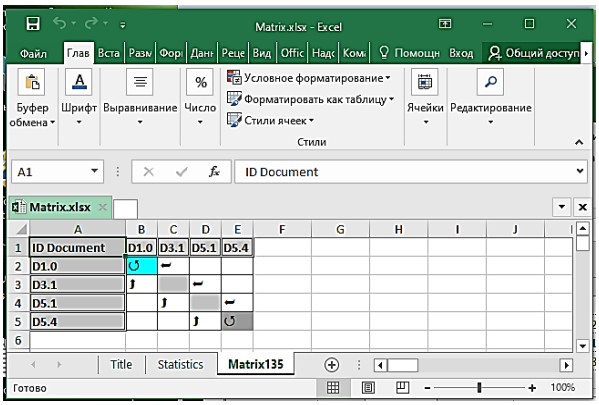

Fig. 2. Constructed Traceability Matrix

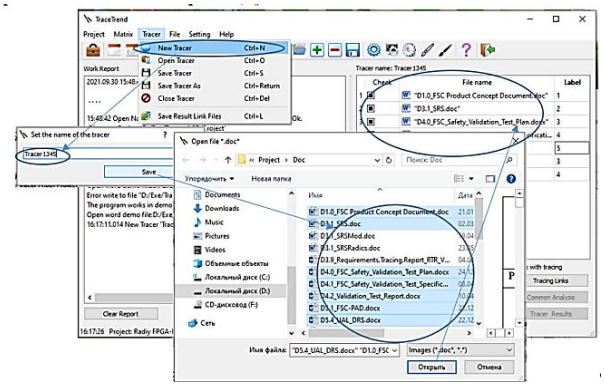

Fig. 3. Test Coverage Verification

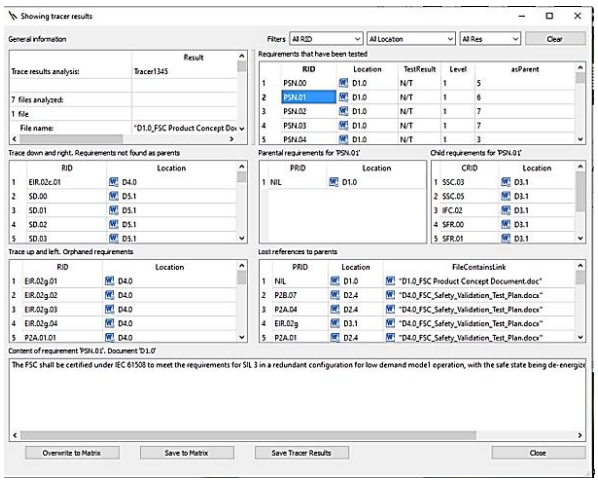

Fig. 4. Reporting and Result Analysis

3.2. Advantages of the TraceTrend Tool

The use of TraceTrend simplifies and improves the requirements traceability process by:

– automating traceability operations, which reduces the likelihood of errors associated with manual data entry or modification;

– reducing the time required to verify project documentation for compliance with requirements, thereby significantly accelerating the development and testing process.

TraceTrend can be particularly useful for small and medium-sized projects where access to large and expensive requirements management systems is limited. Due to its flexibility, TraceTrend serves as an effective tool for analyzing and tracing requirements in resource-constrained projects. The tool can also be used for educational purposes to demonstrate the requirements traceability process and verify documentation compliance with established standards.

To evaluate the efficiency of the developed tool, a pilot study was conducted on a medium-scale project specification (≈240 requirements). The comparison between manual verification and TraceTrend-based analysis is presented in Table 1.

Table 1

Quantitative comparison of manual vs. TraceTrend-based traceability verification

Approach	Verification time (hours)	Undetected traceability issues
Manual review	12	4
TraceTrend	7	1

4. Discussion

The results of the study demonstrate that the formalization of requirements traceability through mathematical models and its implementation in the TraceTrend tool provides significant advantages over manual or semi-automated approaches.

By introducing binary relations, directed graphs, and Boolean matrices into the process of documentation analysis, it becomes possible to not only verify the presence of links but also to assess their logical consistency and completeness. This aspect is crucial in safety-critical projects, where gaps or inconsistencies in traceability may lead to non-compliance with standards such as IEC 61508 or ISO/IEC/IEEE 29148, thereby jeopardizing certification.

A key strength of the TraceTrend approach is its focus on document-oriented projects, where requirements are embedded in textual artifacts such as specifications, design descriptions, and test reports in Word and PDF formats. Unlike heavy ALM platforms (e.g., IBM DOORS or Polarion ALM), which are optimized for enterprise environments but costly and difficult to adapt, TraceTrend offers a lightweight and configurable alternative.

This makes it especially valuable for small-to-medium projects, academic settings, and organizations with limited resources. Furthermore, the ability to integrate formal validation rules into documentation workflows ensures that traceability violations are detected at early stages, reducing rework and improving overall development efficiency.

At the same time, several limitations were identified. First, TraceTrend requires initial configuration of regular expressions for requirement markup, which may demand effort from the project team and domain experts. Although this step is a one-time investment, it represents a barrier for non-technical users compared to systems with built-in parsers and templates. Second, the tool supports only a limited set of document formats (DOCX, PDF), which may restrict its adoption in organizations relying on LaTeX or specialized modeling environments. Third, while TraceTrend performs well in identifying undocumented requirements and coverage gaps, it does not yet fully integrate with broader project management systems, test automation pipelines, or version control repositories.

From a scientific perspective, the results align with recent research trends in the field of requirements engineering. Approaches such as graph-based traceability models, natural language processing for trace link recovery, and ML-based classification of requirement-test pairs demonstrate a shift towards automation and intelligence in traceability management. TraceTrend contributes to this discourse by showing that formal mathematical modeling, even without advanced AI, can provide a robust foundation for ensuring traceability in real-world documentation.

Finally, the incorporation of visualization techniques (interactive dependency graphs, dashboards) would improve the interpretability of analysis results and support project managers in decision-making.

The discussion highlights that TraceTrend bridges a practical gap between the need for formal rigor in traceability analysis and the demand for accessible, low-cost tools in engineering projects. While the tool cannot yet replace enterprise-level platforms, it provides a scalable and extensible framework that can evolve with project needs, making it a valuable addition to the toolbox of requirements engineering.

As shown in Table 2, TraceTrend offers a unique balance of formal validation, low cost, and ease of use, which distinguishes it from existing categories of traceability tools.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Table 2

Comparison of Traceability Tools

Criterion	ALM tools (DOORS, Polarion, Jama)	Test management (Jira/Xray, TestRail)	UML modeling tools	Spreadsheets (Excel)	TraceTrend
Cost / accessibility	High	Medium	Medium	Very low	Low
Ease of deployment	Low	Medium	Medium	High	High
Coverage verification	Strong	Limited to tests	Weak	None	Strong
Formal model validation	No	No	Partial	No	Yes
Suitability for small teams	Low	Medium	Low/Medium	High	High

5. Conclusions

This study has addressed the problem of automating requirements traceability in project documentation for hardware–software systems, with particular emphasis on formal correctness, completeness, and test coverage.

The objective of the research–developing a mathematically grounded and tool-supported approach to requirements traceability–has been successfully achieved. As a result of the research, a formal model of requirements traceability was developed based on set theory, binary relations, directed graphs, and Boolean matrices. The model provides a rigorous framework for representing requirement hierarchies, associations between requirements and documents, traceability links between requirements and test cases. By distinguishing between association (incidence) relations and structural dependency relations, the model ensures semantic clarity and formal consistency.

The proposed model was implemented in the TraceTrend software tool, which enables automated identification of requirements in project documents, construction of traceability matrices, and verification of test coverage. The tool supports hierarchical traceability across multiple documentation levels and allows detection of missing, uncovered, or inconsistently linked requirements at early stages of the project lifecycle.

The obtained results confirm that the integration of formal modeling with automated analysis tools significantly improves the reliability and auditability of requirements management processes. The developed approach contributes to improving documentation quality and supports compliance with functional safety and requirements engineering standards such as IEC 61508 and ISO/IEC/IEEE 29148. Overall, the study demonstrates that mathematically justified traceability models, when combined with lightweight and configurable tooling, provide an effective solution for managing requirements in safety-critical and resource-constrained engineering projects.

Future research will focus on extending the TraceTrend tool to support additional document formats, in order to broaden its applicability in academic and industrial environments including integration with algebraic methods of requirement engineering and verification [23, 24].

Contribution of authors: article concept, problem formulation, writing of the original manuscript – **Oleg Odarushchenko;** development of the mathematical model – **Olena Odarushchenko;** concept of the traceability tool, analysis and validation of results, software testing, and manuscript review – **Oleksii Striuk, Viacheslav Shamanskiy;** development of the TraceTrend tool, preparation of demonstration examples – **Petro Hroza.**

Conflict of Interest

The authors declare that they have no conflict of interest concerning this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

This study was conducted without financial support.

Data Availability

The manuscript has no associated data.

Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence methods while creating the presented work.

All the authors have read and agreed to the published version of this manuscript.

References

1. *IEC 61508:2010. Functional safety of electrical/electronic/programmable electronic safety-related systems*. Published. 2010 – 04. IEC Standards, 2010. 594 p.

2. *ISO/IEC/IEEE 29148. Systems and software engineering – Life cycle processes – Requirements engineering*. International Organization for Standardization / International Electrotechnical Commission / Institute of Electrical and Electronics Engineers, 2018. 134 p.

3. Fucci, D., Unterkalmsteiner, M., Fernández, D.M., Gorschek, T., & Wagner, S. When traceability goes awry: An industrial experience report. *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Pittsburgh, PA, USA, May 2022, pp. 41–50. DOI: 10.1145/3510457.3513022.

4. Sommerville, I. *Software Engineering*, 10th ed., USA, Pearson Education, 2016. 812 p. ISBN: 978-0-13-394303-0.

5. Wiegers, K., & Beatty, J. *Software Requirements*, 3rd ed. Redmond, WA, USA, Microsoft Press, 2013. 648 p. ISBN: 978-0-7356-7966-5.

6. Martins, L. E. G., & Gorschek, T. (Eds.) *Requirements Engineering for Safety-Critical Systems.* Gistrup, Denmark: River Publishers, 2021. 218 p. ISBN: 978-87-7022-427-7.

7. Ramesh, B., & Jarke, M. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, 2001. vol. 27, no. 1, pp. 58–93. DOI: 10.1109/895989.

8. Ratiu, C., Mayr-Dorn, C., Assunção, W., & Egyed, A. Taming cross-tool traceability in the wild. *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, 2023, pp. 233–243. DOI: 10.1109/RE57278.2023.00031.

9. Ilyas, I., Hafees, Y., Hussain, S. Jammal, M., & Toure, I. K. Optimized change management process through semantic requirements and traceability analysis tool. *Journal of Engineering*, 2025, article ID 2296387. 17 p. DOI: 10.1155/je/2296387.

10. Mucha, J., Kaufmann, A., & Riehle, D. A systematic literature review of pre-requirements specification traceability. *Requirements Engineering*, 2024, vol. 29, pp. 119–141. DOI: 10.1007/s00766-023-00412.

11. Zhao, Z., Su, Y., Li, Y., Zou, Y., Li, R., & Zhang, R. A survey on self-supervised graph foundation models: Knowledge-based perspective. *Journal of LaTeX Class Files*, 2020, vol. 18, iss. 9, pp. 1–12. DOI: 10.48550/arXiv.2403.16137.

12. Pauzi, Z., & Capiluppi, A. Applications of natural language processing in software traceability: A systematic mapping study. *Journal of Systems and Software*, 2023, vol. 198, article 111616. DOI: 10.1016/j.jss.2023.111616.

13. Elangovan, G. R., Sujitha, V., Sasirekha, V., Prisca Mary, J., V.R.R., & Vengatesh, T. Machine Learning Algorithms in the Detection of Pattern System using Algorithm of Textual Feature Analysis and Classification. *Journal of Neonatal Surgery*, 2025, vol. 14(14s), pp. 66–74. DOI: 10.63682/jns.v14i14S.3430.

14. Lin, J., Liu, Y., Zeng, Q., Jiang, M., & Cleland-Huang, J. Traceability Transformed: Generating more Accurate Links with Pre-Trained BERT Models. *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*, 2021, Madrid, Spain, May, pp. 324–335. DOI: 10.1109/ICSE43902.2021.00040.

15. Guo, J. L. C., Steghöfer, J.-P., Vogelsang, A., & Cleland-Huang, J. Natural Language Processing for Requirements Traceability. *Handbook of Natural Language Processing for Requirements Engineering*, 2024. DOI: 10.48550/arXiv.2405.10845.

16. Maro, S., Steghöfer, J.-P., Knauss, E., Horkoff, J., Kasauli, R., Wohlrab, R., Korsgaard, J. L., Wartenberg, F., Strøm, N. J., & Alexandersson, R. Managing traceability information models: Not such a simple task after all? *IEEE Software*, 2021, vol. 38, iss. 5, pp. 101–109. DOI: 10.1109/MS.2020.3020651.

17. Chandrika, A. R. R. N. Test Case Management in Agile Software Development Using Jira. *International Journal for Multidisciplinary Research (IJFMR)*, 2024, vol. 4, iss. 4, article no. 20560. DOI: 10.36948/ijfmr.2022.v04i04.20560.

18. Yoo, I., Park, H., Lee, S.-W., & Ryu, K.-Y. Building traceability between functional requirements and component architecture elements in embedded software using structured features. *Applied Sciences*, 2024, vol. 14, no. 23, aricle. no. 10796, pp. 1–23. DOI: 10.3390/app142310796.

19. Medvedík, M., & Ždánsky, J. Safety PLC Programming Based on UML Statechart", *2020 29th IEEE International Conference on Emerging eLearning Technologies and Applications (ELEKTRO)*, 2020, pp. 1–5. DOI: 10.1109/ELEKTRO49696.2020.9130307.

20. Krüger, G. How to Create a Requirements Traceability Matrix – with Examples. *Perforce Blog*, 26 June 2025. Available at: https: //www.perforce.com/blog/ alm/how-create-traceability-matrix (accessed: 25 August 2025).

21. Bonner, M., Zeller, M., Schulz, G., Beyer, D., & Olteanu, M. Automated traceability between requirements and model-based design. *Proceedings of the REFSQ-2023 Workshops* (CEUR Workshop Proceedings, 2023, vol. 3378, pp. 1–7. Available at: https://ceur-ws.org/Vol-3378/PT-paper3.pdf. (accessed: 25 August 2025).

22. Dai, P., Yang, L., Wang, Y., Jin, D., & Gong, Y. Constructing Traceability Links between Software Requirements and Source Code Based on Neural Networks. *Mathematics*, 2023, vol. 11, article no. 315. DOI: 10.3390/math11020315.

23. Letychevskyi, O., Peschanenko, V., Kharchenko, V., Volkov, A., & Odarushchenko, O. Modeling method for development of digital system algorithms based on programmable logic devices. *Cybern.*

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Syst. Analysis, 2020, vol. 56, no. 5, pp. 710–717. DOI: 10.1007/s10559-020-00289-8.

24. Letychevskyi, O., Odarushchenko, O., Peschanenko, V., Kharchenko, V., & Moskalets, V. Insertion Semantics of VHDL as Electronic Design Languge. *Cybern. Syst. Analisys*, 2022, vol. 58, pp. 289–298. DOI: 10.1007/s10559-022-00461-2.

Received 14.06.2025, Received in revised form 12.10.2025
Accepted date 17.11.2026, Published date 08.12.2025

АВТОМАТИЗАЦІЯ ТРАСУВАННЯ ВИМОГ У ПРОЄКТНІЙ ДОКУМЕНТАЦІЇ ЗА ДОПОМОГОЮ ІНСТРУМЕНТА TRACETREND: МОДЕЛЬ, ДИЗАЙН, ЗАСТОСУВАННЯ

О. М. Одарущенко, О. Б. Одарущенко, О. Ю. Стрюк, В. О. Шаманський, П. М. Гроза

У статті розглянуто питання трасування вимог у контексті інженерії програмного забезпечення та системної інженерії, зокрема, в галузі розробки програмно-апаратних засобів для критичних систем. **Об'єктом** дослідження є процес трасування вимог у проєктній документації, а **предметом** – математичне моделювання та інструментальна реалізація цього процесу. **Метою досліджень є** розробка математично обґрунтованого підходу до трасування вимог у проєктній документації, що забезпечує автоматичне виявлення та перевірку вимог, побудову їхніх ієрархічних зв'язків, аналіз покриття тестами та виявлення порушень трасувальних зв'язків, з урахуванням стандартів функціональної безпечності та вимог до якості документації.Для досягнення мети дослідження були застосовані методи теорії множин, бінарних відношень, орієнтованих графів, а також методи програмної реалізації та тестування. У результаті дослідження була розроблена математична модель трасування вимог, яка включає основні множини (документи, ідентифікатори вимог, тестові випадки), бінарні відношення (належність, спадкоємність, покриття тестами) та функцію результату тестування. Модель була реалізована в інструменті **TraceTrend**, що автоматизує процес трасування вимог у проєктних документах, перевіряє покриття тестами та виявляє порушення умов трасування. **TraceTrend** дозволяє ефективно працювати з ієрархічними зв'язками між вимогами, тестами та іншими елементами документації, що забезпечує високий рівень точності та відповідності стандартам безпеки, таким як IEC 61508, ISO 26262. Основним результатом є створення інструменту для автоматизованого аналізу проєктної документації, що значно знижує ймовірність помилок та дозволяє забезпечити цілісність і повноту вимог. Також було виявлено, що використання **TraceTrend** зменшує час, необхідний для перевірки документації, та підвищує ефективність процесу тестування. Висновки дослідження свідчать, що математичне моделювання трасування вимог у поєднанні з інструментальною реалізацією може значно покращити управління вимогами в проєктах, особливо в критичних або високонадійних системах, та забезпечити відповідність міжнародним стандартам.

Ключові слова: трасування вимог; математичне моделювання; інструмент TraceTrend; бінарні відношення; покриття тестами; стандарти безпеки.

Одарущенко Олег Миколайович – д-р техн. наук, проф., проф. каф. інформаційних систем та технологій , Полтавський державний аграрний університет, Полтава, Україна.

Одарущенко Олена Борисівна – канд. техн. наук, доц., доц. каф. інформаційних систем та технологій, Полтавський державний аграрний університет, Полтава, Україна.

Стрюк Олексій Юрійович – канд. техн. наук, доц., старш. наук. співроб., Науково-виробниче підприємство «Радікс», Кропивницький, Україна.

Шаманський Вячеслав Олегович – старш. інженер-програміст, Науково-виробниче підприємство «Радікс», Кропивницький, Україна.

Гроза Петро Миколайович – канд. техн. наук, доц., доц. каф. інформаційних систем і технологій, Військовий інститут телекомунікації та інформатизації імені Героїв Крут, Київ, Україна.

Oleg Odarushchenko – Doctor of Technical Sciences, Professor, Professor of the Department of Information Systems and Technologies Poltava State Agrarian University, Poltava, Ukraine,
e-mail: odarushchenko@gmail.com, ORCID: 0000-0003-3933-9637.

Olena Odarushchenko – Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Information Systems and Technologies, Poltava State Agrarian University, Poltava, Ukraine,
e-mail: elena.odarushchenko@gmail.com, ORCID: 0000-0002-2293-2576.

Oleksii Striuk – Candidate of Technical Sciences, Associate Professor, Lead Verification Engineer, RadICS LLC, Kropyvnytskyi, Ukraine, e-mail: oleksii.striuk@radics-ua.com, ORCID: 0009-0009-2237-1623.

Viacheslav Shamanskyi – Senior, Software Engineer, RadICS LLC, Kropyvnytskyi, Ukraine,
e-mail: viacheslav.shamanskyi@radics-ua.com, ORCID: 0000-0001-8228-282X.

Petro Hroza – Candidate of Technical Sciences, Associate Professor of the Department of Information Systems and Technologies, Heroes Krut Military Institute of Telecommunications and Informatization, Kyiv, Ukraine,
e-mail: groza@ukr.net, ORCID: 0000-0001-5308-4728.