Security of Infocommunication Systems and Internet of Things

2025 Vol 3, No 1

https://doi.org/10.31861/sisiot2025.1.01003

Received 03 May 2025; revised 19 May 2025; accepted 08 June 2025; published 30 June 2025

A Methodological Approach to Auditing Software Engineering Practices in the Energy Sector

Ihor Liutak[1,*] and Zinoviy Liutak[2]

[1]Software Engineering Department, Ivano-Frankivsk National Technical University of Oil and Gas, Ivano-Frankivsk, Ukraine
[2]Information and Measurement Technologies Department, Ivano-Frankivsk National Technical University of Oil and Gas, Ivano-Frankivsk, Ukraine
*Corresponding author (E-mail: ihor.liutak@nung.edu.ua)

ABSTRACT The growing complexity of software systems in the energy sector, particularly those involved in the management of distributed and renewable energy resources, requires the introduction of structured and domain-specific auditing methodologies. Ensuring the reliability, safety, and security of software products in this context is critical due to the increasing dependence of industrial and energy infrastructures on automated and software-driven solutions. This paper proposes a comprehensive methodological approach to auditing software engineering practices tailored to the needs of the energy sector. The developed methodology is based on an integrated audit model that defines Process, Product, and Safety and Security layers to enable a holistic and systematic evaluation. Furthermore, it incorporates a structured audit process aligned with quality management principles, covering all essential stages from planning to follow-up. A key feature of the approach is the mathematical formalization of audit activities, which includes models for estimating effort, measuring audit coverage, analysing nonconformities, and evaluating process maturity. These models enhance the objectivity and analytical rigor of audits, enabling organizations to quantify and compare results across projects and audit cycles. The proposed methodology was developed based on a thorough analysis of international standards, including ISO/IEC 12207, ISO/IEC 25010, IEC 61508, IEC 62443, and ISO 9001, and aims to bridge the gap between general software engineering requirements and domain-specific needs related to functional safety, cybersecurity, and operational reliability. The results of this research contribute to the advancement of audit methods in the field of software engineering and provide a scientifically substantiated and practically oriented tool for improving the quality, security, and compliance of software systems used in the energy sector.

KEYWORDS software audit, energy sector, software engineering, functional safety, cybersecurity.

I. INTRODUCTION

The energy sector increasingly relies on complex software systems to monitor, control, and optimize industrial processes. As these systems become more integrated with critical infrastructure and data-driven operations, ensuring their reliability, security, and quality becomes paramount. This necessity places new demands on software development practices, especially within industrial domains where system failures can result in significant economic, environmental, or safety consequences. Consequently, the role of software engineering audits is systematic evaluations of development processes and outcomes. It has gained importance as a means to enforce compliance with standards and improve software quality. In industrial settings, software audits must address not only traditional aspects of code quality and process maturity but also domain-specific requirements such as interoperability with SCADA systems [1], compliance with regulatory frameworks such ISO/IEC 12207 [2, 3], and robustness in harsh operating environments [4]. However, the lack of unified audit methodologies tailored to the energy sector often leads to inconsistent or incomplete assessments. This highlights a pressing need for a structured and standardized approach to auditing that integrates software engineering principles with the operational realities of industrial energy systems.

The digitalization of the energy sector has introduced new challenges in maintaining the quality, reliability, and compliance of software systems that operate within mission-critical environments. As software becomes a key enabler of energy generation, distribution, and consumption optimization, flaws in its development process may lead to operational inefficiencies, cybersecurity vulnerabilities, or even catastrophic failures. Despite the critical nature of these systems, many energy enterprises continue to lack formalized and domain-specific mechanisms for auditing software development processes. Therefore, the development of a methodological approach to auditing is aligned with international software engineering standards and tailored to the needs of the energy sector. It remains a timely and necessary scientific endeavour.

The purpose of this work is to develop and present a methodological approach to auditing software engineering practices in the industrial energy sector, with the aim of enhancing quality assurance, ensuring regulatory compliance, and improving the reliability of software-intensive systems.

The main objectives of this research are to examine existing software audit standards and approaches relevant to industrial and energy-sector applications, and to identify the specific risks and challenges that arise in the development of software for critical energy systems. Based on this analysis, the study aims to develop a clear and structured methodology for auditing software engineering practices, tailored to the needs of the energy sector. The proposed method is intended to support practical

implementation through example-based demonstration and to offer concrete recommendations for its integration into existing quality management and compliance frameworks used by energy companies.

Amoo et al. [5] developed a multicriteria framework for assessing energy audit software used in residential applications. Although the main context of their study is different, the proposed evaluation criteria – including usability, accuracy, and regulatory compliance provide useful ideas for structuring software audits. The emphasis on aligning software functionality with regulatory and end-user needs is particularly applicable to auditing industrial energy sector software, where similar concerns about compliance and operational effectiveness exist. However, the scope of the work is limited to residential energy systems and does not consider the complexities of industrial software development processes. Abbas et al. [6] examined secure software development practices with a focus on ensuring the reliability and integrity of critical software systems. Their research highlights how process audits and adherence to secure coding standards contribute to reducing vulnerabilities, a priority that is also crucial in the energy sector, where software failures can lead to major disruptions. Although their study targets the banking industry, the proposed secure development lifecycle practices and audit mechanisms are broadly applicable to other domains, including energy. However, it does not discuss challenges such as interoperability with industrial systems or operating in harsh physical environments. Jena [7] investigated how blockchain technology could enhance auditability and transparency in accounting processes, using a multi-criteria decision-making framework. The study introduces concepts such as immutable records and decentralized validation, which could be adapted to improve the traceability and reliability of software audits in the energy industry. While the research focuses on financial applications, its findings suggest potential benefits for auditing software engineering processes, particularly in enhancing trust and integrity. Nevertheless, domain-specific requirements such as compliance with industrial software standards and environmental robustness were not addressed. Tucker [8] studied the influence of software development methodologies on project success in the automotive industry. The research provides insights into how process choices affect the quality and reliability of complex software products. These findings are relevant to the energy sector, where selecting appropriate development models and ensuring process adherence are critical for producing dependable software. However, the work does not focus on software auditing itself or on how methodologies are verified or validated in regulated industrial environments. Diyab et al. [9] explored the use of engineered prompts in ChatGPT to support educational assessment in software engineering. While the primary focus is on academic applications, the research demonstrates how AI-driven tools can be used to systematically evaluate knowledge and processes. This is relevant to software auditing, where automated reasoning and assessment could enhance the objectivity and efficiency of audit procedures. However, the study does not address the industrial context or the requirements of mission-critical software, which limits its direct applicability to energy sector auditing. Ganapathy and Sampath [10] conducted a systematic literature review on regulatory and security compliance in cloud-based software ecosystems. Their research synthesizes existing standards, frameworks, and challenges related to ensuring compliance in distributed and virtualized environments. These insights are valuable for energy sector software auditing, as many modern industrial applications increasingly rely on cloud technologies. Nonetheless, the reviewed works mainly focus on cloud computing and do not fully address software quality assurance in embedded or real-time systems typically found in energy infrastructure. Sholihin and Salman [11] introduced OSCAT, an automated tool designed for auditing against CIS Benchmarks. This tool showcases the potential of automation in the audit process, particularly in enhancing consistency and reducing human error. For the energy sector, adopting similar automation strategies could significantly improve the reliability and repeatability of software audits. However, OSCAT is specialized for cybersecurity configurations rather than software engineering process audits, limiting its immediate applicability to broader software development quality assessments. Terragni et al. [12] discussed future directions in AI-driven software engineering, focusing on how artificial intelligence can automate and optimize various stages of the software lifecycle. Their work highlights promising trends that could be integrated into audit methodologies to enhance coverage, predictive capabilities, and decision support. Nevertheless, while AI offers transformative potential, the paper does not provide specific strategies or guidelines for applying AI techniques within audit frameworks for industrial software, leaving a gap in practical implementation for energy-critical systems.

Recent research highlights notable advancements in the field of software auditing and quality assurance. Various approaches have been proposed to enhance regulatory compliance, improve process transparency, and introduce automation into audit activities. Secure development practices, automated assessment tools, and the integration of AI-driven technologies have been recognized as effective strategies to increase the reliability and consistency of audits across multiple domains. These developments demonstrate a growing recognition of the importance of systematic and standardized approaches to software evaluation. However, the majority of existing solutions and methodologies remain largely focused on general-purpose software, cybersecurity audits, and cloud-based environments. As a result, they often overlook the unique challenges associated with software developed for industrial and energy sectors. In such domains, software systems must not only meet conventional quality criteria but also ensure seamless interoperability with control and monitoring infrastructure, demonstrate robustness in harsh operational conditions, and comply with strict domain-specific standards and regulations. Furthermore, while automation and artificial intelligence are increasingly discussed in the context of software assessment, their application to audit methodologies designed for mission-

critical industrial software remains insufficiently explored. These limitations underline the need for a dedicated methodological approach to auditing software engineering practices within the energy sector. Such an approach should combine the principles of standardization, verification, validation, and unification, while also addressing the particular operational and regulatory requirements of energy-related software. Filling this gap is essential to achieving higher levels of audit consistency, improving regulatory alignment, and enhancing the dependability and safety of software solutions deployed in critical energy infrastructures.

II. REVIEW AND ANALYSIS OF INTERNATIONAL STANDARDS RELEVANT TO SOFTWARE AUDITING IN THE ENERGY SECTOR

Software engineering standards provide a universal basis for ensuring consistency, quality, and traceability throughout the software lifecycle. In the context of auditing software engineering practices in the energy sector, they help establish what processes should exist, how they should be performed, and what criteria must be used to verify outcomes. The following standards define key methodologies, models, and frameworks that are foundational for building an audit methodology.

ISO/IEC 12207 is the primary international standard that defines the processes involved in the software lifecycle, from conception to retirement [13]. It classifies processes into primary life cycle processes (acquisition, supply, development, operation, and maintenance), supporting processes (documentation, configuration management, quality assurance, verification, validation, etc.), and organizational processes (management, infrastructure). This standard is essential for audits because it defines what processes should exist in any compliant software organization. Auditors can use ISO/IEC 12207 as a checklist for verifying process presence, adequacy, and outcomes at each lifecycle stage. ISO/IEC 15288, although broader, complements ISO/IEC 12207 by covering the system lifecycle, not just software [14]. This is especially important for the energy sector, where software is tightly integrated with hardware and physical infrastructure (SCADA, sensors, controllers). ISO/IEC 15288 defines system-level processes such as stakeholder requirements definition, system requirements definition, architectural design, integration, verification, and validation. These processes provide auditors with models for assessing how software interacts within the broader energy system and how well system-level requirements are traced and satisfied. IEEE 730 focuses specifically on software quality assurance (SQA) plans [15]. It offers guidance on the preparation of SQA plans, detailing what should be planned, monitored, and controlled during the software development process. For audit purposes, IEEE 730 is valuable because it formalizes the expectations for SQA activities and provides criteria against which the implementation of quality assurance procedures can be assessed. IEEE 1012 defines the verification and validation (V&V) processes for software [16]. It specifies how independent assessment activities should be conducted to determine whether software products and processes meet their specified requirements and intended uses. In audits, this standard provides a methodological basis for evaluating the adequacy and independence of V&V activities, which is critical for critical systems, including those in the energy sector. ISO/IEC 25010 introduces the quality model, defining software product quality attributes such as functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability [17]. This standard does not define process models, but it is essential in audits for defining the criteria and metrics used when evaluating the quality of software products produced by engineering processes. It helps link process quality with product quality outcomes. In summary, these standards together offer auditors a structured framework for evaluating software engineering processes and outcomes. ISO/IEC 12207 and ISO/IEC 15288 establish comprehensive lifecycle process models that define how software and systems should be developed and maintained [14]. IEEE 730 and IEEE 1012 provide methodologies for assuring and verifying the quality of both processes and products, while ISO/IEC 25010 offers criteria for assessing software product quality [15, 16]. Collectively, these standards form the foundation for audit methodologies aimed at ensuring compliance, traceability, and reliability in software engineering practices, particularly in critical sectors such as energy. The key characteristics and audit-relevant features of each standard are summarized in Table 1.

TABLE 1. Features of General Software Engineering and Lifecycle Standards.

Standard	Scope	Proposes / Defines	Relevance for Audit
ISO/IEC 12207	Software lifecycle processes	Process model and definitions	Process existence, adequacy, and completeness checks
ISO/IEC 15288	System lifecycle processes	System-level process model	System integration and requirements traceability
IEEE 730	Software quality assurance plans	SQA plan structure and content	Planning and assessment of QA activities
IEEE 1012	Verification and validation processes	V&V process methodology	Independent assessment of software conformity
ISO/IEC 25010	Software product quality attributes	Quality model (attributes, metrics)	Evaluation criteria for product audit

While general software engineering standards provide a comprehensive foundation for defining lifecycle processes and product quality attributes, they are not sufficient by themselves when it comes to software intended for critical energy infrastructure. In such environments, software must satisfy strict requirements related to safety, security, and overall process capability to

ensure operational reliability and regulatory compliance. To address these needs, a range of specialized standards has been developed that extend beyond general software lifecycle management. This group includes standards for functional safety, cybersecurity for industrial control systems, and process maturity assessment. The following analysis highlights how these standards contribute to the development of audit methodologies suited for software engineering practices in the energy sector.

IEC 61508 serves as the foundational standard for functional safety of electrical, electronic, and programmable electronic safety-related systems [18]. It defines a risk-based approach to determining Safety Integrity Levels (SIL) and provides detailed guidance on the lifecycle processes required to achieve the desired level of safety. For software auditors, IEC 61508 offers criteria for assessing whether software development processes have adequately addressed safety planning, requirements specification, design, implementation, verification, validation, and maintenance. It emphasizes independence in verification and documentation as part of demonstrating compliance with safety objectives. IEC 62443 focuses on the security of industrial automation and control systems, which are central to energy sector operations [19]. This standard provides a comprehensive framework for addressing cybersecurity across system lifecycle phases, including software development. It defines security levels, security policies, secure coding practices, and validation procedures necessary to mitigate cybersecurity risks. For auditing purposes, IEC 62443 is essential for verifying that software engineering processes incorporate security requirements, conduct threat modelling, and apply secure development practices suitable for critical infrastructure. ISO/IEC 33001–33099 (formerly ISO/IEC 15504 or SPICE) establishes a process assessment framework that allows organizations to evaluate the capability and maturity of their software processes [20]. It defines process attributes and measurement scales, enabling systematic assessment of how well processes are defined, managed, and optimized. For software audits, these standards provide a structured approach to measuring process performance and identifying areas for improvement. CMMI (Capability Maturity Model Integration), although not an ISO standard, is a globally recognized model for process improvement [21]. It offers detailed guidelines for assessing and improving processes across various maturity levels, from initial (ad hoc) to optimizing (continuous improvement). In the context of audits, CMMI serves as a reference model for evaluating process maturity, organizational readiness, and adherence to best practices in software development. Collectively, these standards address critical aspects of software auditing in the energy sector. IEC 61508 and IEC 62443 offer domain-specific requirements for safety and security, while ISO/IEC 330xx and CMMI provide frameworks for assessing process capability and improvement. Together, they enable auditors to evaluate not only whether processes exist, but also whether they are sufficiently robust, safe, secure, and capable of delivering high-quality and compliant software. The main features and audit-relevant contributions of these standards are summarized in Table 2.

TABLE 2. Features of Functional Safety and Process Assessment Standards.

Standard	Scope	Proposes / Defines	Relevance for Audit
IEC 61508	Functional safety of industrial systems	Safety lifecycle model, SIL, verification & validation criteria	Assessment of safety-related development and compliance
IEC 62443	Cybersecurity for industrial control systems	Security levels, secure development and validation	Audit of cybersecurity practices and secure software development
ISO/IEC 33001–33099	Software process assessment framework	Process attributes, assessment models	Evaluation of process capability and maturity
CMMI	Process improvement and maturity model	Process maturity levels, best practices	Audit of process organization, performance, and improvement efforts

III. METHODOLOGY PRINCIPLES AND AUDIT MODEL

The analysis of international standards reveals that effective auditing of software engineering practices in the energy sector requires a comprehensive and systematic approach. Given the critical role of software in ensuring the safety, reliability, and efficiency of industrial energy systems, the audit methodology must go beyond simple checklist-based evaluations. It should integrate principles of process maturity assessment, verification and validation, functional safety, and cybersecurity, while ensuring alignment with established software engineering lifecycle models. Based on the reviewed standards, the proposed audit methodology is structured around key principles that reflect best practices in quality assurance and process evaluation. These include process transparency, traceability of requirements, conformance to safety and security standards, and measurable assessment criteria for product quality. The methodology is designed to assess both the software development processes and the resulting products, ensuring that they meet applicable regulatory requirements and industry expectations. The audit model adopts a layered structure that reflects the various levels of assessment necessary for comprehensive evaluation. At the foundational level, the existence and adequacy of defined software processes are assessed, drawing from lifecycle standards such as ISO/IEC 12207 and ISO/IEC 15288 [3]. At the next level, the focus shifts to the implementation and effectiveness of quality assurance, verification, and validation procedures, following guidelines from IEEE 730 and IEEE 1012. The highest level involves evaluation of the software product itself, using ISO/IEC 25010 quality attributes to judge functional suitability, reliability, maintainability, and security. Additional layers address domain-specific concerns such as functional safety (IEC 61508) and cybersecurity (IEC 62443), ensuring that the

audit reflects the unique requirements of industrial energy software. This structured approach provides a clear framework for auditors, supporting consistent, objective, and repeatable assessment processes. By synthesizing the requirements of international standards into an integrated audit model, the methodology aims to bridge the gap between abstract standards and practical auditing tasks in complex, safety- and security-critical environments such as the energy sector.

In complex and safety-critical industries such as energy, software auditing cannot rely on simple or isolated evaluation techniques. Instead, a comprehensive approach is required that considers multiple dimensions of software engineering practices. To achieve this, the proposed audit methodology is structured into layers, each designed to focus on a specific aspect of the software development and product lifecycle. The layered approach offers several important advantages. First, it promotes clarity and structure by dividing the audit into logical areas of focus. This allows auditors to work systematically, without overlooking important factors. Second, it supports traceability and completeness, as each layer is aligned with specific international standards and audit objectives. Third, layering enables flexibility and scalability: auditors can adjust the depth of analysis in each layer depending on the criticality of the software and the context of the audit. Finally, this approach ensures integration of domain-specific requirements, particularly in relation to safety and security, which are essential for energy sector applications.

Process Audit Layer. This foundational layer evaluates whether software lifecycle processes exist, are adequately documented, and effectively implemented within the organization. Using standards such as ISO/IEC 12207 and ISO/IEC 15288, auditors verify process completeness, clarity, and conformity to established lifecycle models [3]. Additional process maturity frameworks such as ISO/IEC 330xx and CMMI are employed to assess how consistently and efficiently these processes are managed and improved. The outcome of this audit layer is an assessment of process capability, maturity, and potential gaps that may affect software quality and compliance.

Product Audit Layer. At this layer, the focus shifts from process quality to evaluating the software products themselves. Auditors use ISO/IEC 25010 quality attributes such as functional suitability, reliability, performance efficiency, maintainability, and security as criteria for assessment. Product audits involve systematic testing, verification, and validation according to IEEE 1012 and ISO/IEC/IEEE 29119 guidelines. Auditors determine whether the final software meets its specified requirements, performs as expected in operational conditions, and complies with quality criteria defined by stakeholders and standards.

Safety and Security Audit Layer. Recognizing the critical nature of software in energy systems, this layer addresses domain-specific concerns such as functional safety and cybersecurity. Safety audits use IEC 61508 to assess whether adequate measures have been taken throughout the software lifecycle to identify, control, and mitigate safety-related risks. Security audits utilize IEC 62443 to evaluate how well cybersecurity requirements are integrated into the software development process, including secure coding practices, threat modelling, security testing, and vulnerability management. This layer ensures the audited software not only functions correctly but also remain safe and secure under real-world conditions.

The layered structure of the audit model ensures a systematic and thorough assessment of software engineering practices. Each layer addresses a specific dimension from organizational processes and software product quality to critical domain-specific aspects such as safety and cybersecurity. This approach not only promotes clarity and completeness but also allows auditors to adapt the methodology to the unique needs of the energy sector, where reliability, risk mitigation, and regulatory compliance are of utmost importance. To illustrate how the model integrates general audit principles with the specific requirements of industrial energy applications, the key features of the proposed approach and their domain relevance are summarized in Table 3.

TABLE 3. Features of the Proposed Audit Model and Its Specific Adaptation for the Energy Sector.

Model Feature	Specific Aspects for the Energy Sector
Layered structure (Process, Product, Safety and Security)	Ensures comprehensive audit coverage, including critical operational aspects
Process compliance assessment (ISO/IEC 12207, ISO/IEC 15288, ISO/IEC 330xx, CMMI)	Verifies process maturity and traceability essential for regulated and mission-critical software
Product quality evaluation (ISO/IEC 25010)	Focus on reliability, maintainability, and security for long-term, continuous operation
Safety assessment (IEC 61508)	Mandatory consideration of risk reduction, Safety Integrity Levels (SIL), and independent verification
Cybersecurity assessment (IEC 62443)	Focus on secure software development and resilience against cyber threats in industrial control environments
Adaptability and scalability of audit depth	Supports different levels of criticality (e.g., SCADA vs non-critical supporting software)
Alignment with organizational QMS and regulatory requirements	Facilitates integration with certifications and inspections relevant to energy sector regulations

IV. AUDIT PROCESS STAGES

While the layered audit model defines what should be evaluated, it is equally important to define how the audit should be conducted. The audit process itself must follow a structured set of stages to ensure consistency, traceability, and completeness. Each stage serves a specific purpose and

contributes to achieving the overall audit objectives. The proposed audit methodology includes the following key stages: Planning, Execution (Assessment), Analysis and Reporting, and Follow-up and Improvement Recommendations.

Planning. The audit process begins with thorough planning, which defines the scope, objectives, and criteria of the audit. During this stage, auditors identify the target software systems, applicable standards, and specific organizational and regulatory requirements. Planning also involves assembling the audit team, determining audit methods (document review, interviews, observations, testing), and scheduling audit activities. Clear and detailed planning ensures that the audit is aligned with organizational goals and that all critical areas will be covered.

Execution (Assessment). The execution stage involves gathering and analysing evidence to assess the conformity and effectiveness of software engineering practices. This stage is conducted in alignment with the audit model layers:

Process Audit: Auditors review documented processes, procedures, and records. They assess process completeness, adherence, and maturity using models such as ISO/IEC 12207 and ISO/IEC 330xx [13, 20].

Product Audit: Auditors verify whether the software meets specified quality attributes, using ISO/IEC 25010 as a reference. This may include review of test results, validation reports, and product documentation [17].

Safety and Security Audit: Auditors assess how well safety and security requirements are integrated into the development process and whether risk mitigation measures comply with IEC 61508 [18].

During this stage, objective evidence is collected through document analysis, interviews with project teams, and observation of development and testing activities.

Analysis and Reporting. Following the assessment, the gathered data is analysed to identify nonconformities, weaknesses, and opportunities for improvement. The findings are then compiled into an audit report. The report should present:

1. Audit objectives and scope;
2. Summary of methods and evidence;
3. Findings for each audit layer (process, product, safety/security);
4. Assessment of compliance with standards;
5. Recommendations for corrective and preventive actions.

The report must be clear, objective, and well-structured, serving as a formal record of the audit results.

Follow-up and Improvement Recommendations. The final stage focuses on follow-up activities. Auditors should verify that corrective actions have been planned and implemented to address identified nonconformities. Additionally, auditors may provide guidance for improving process maturity and strengthening safety and security practices. This stage ensures that the audit contributes not only to compliance verification but also to the continuous improvement of software engineering practices within the organization.

While follow-up typically requires fewer resources compared to earlier stages, its significance should not be underestimated. Effective follow-up consolidates the outcomes of the audit and directly influences long-term process improvements. Based on the authors' experience and observations from auditing software engineering practices in industrial and energy-related projects, the distribution of effort across the audit stages tends to follow a stable pattern. Planning and analysis activities require moderate time and focus, while assessment activities consume the largest share of effort due to their complexity and the necessity of comprehensive evidence collection and verification. To illustrate this empirically derived observation, Fig. 1 presents the approximate distribution of effort across the main audit stages. The data reflects typical audits conducted in critical industrial contexts, where assessment plays the central role in ensuring process integrity and product compliance. Although follow-up represents a smaller proportion in terms of effort, it remains an essential stage, closing the audit loop and promoting continual improvement within audited organizations.

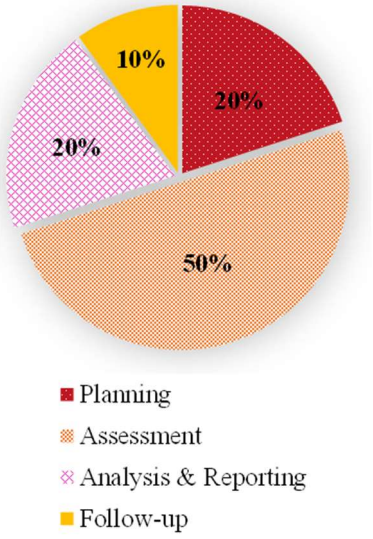

FIG. 1. Approximate Effort Distribution Across Audit Stages.

While Fig. 2 presents the general distribution of effort across all audit stages, it is important to recognize that the emphasis and resource allocation may vary significantly depending on the criticality of the software being audited. In the energy sector, software systems range from non-critical support applications to mission-critical control and safety-related systems. This distinction has a direct impact on the audit approach and effort distribution. Based on practical experience and observations from conducted audits, the allocation of audit effort shifts noticeably between critical and non-critical software projects. For critical systems, such as those involved in SCADA control or safety shutdown functions, auditors tend to dedicate significantly more time to the planning and assessment stages. In such cases, more detailed planning is required to understand regulatory requirements and define adequate audit scopes. Furthermore, assessment activities become more intensive, as verifying compliance with functional safety (IEC 61508) and cybersecurity (IEC 62443) standards demands deeper analysis and more extensive evidence collection. In contrast, audits of non-critical

software typically allow for a more streamlined approach, where relatively less effort is needed for planning and assessment, and more focus is placed on analysis, reporting, and providing improvement recommendations. To reflect these findings and to offer a visual comparison of audit effort allocation, Fig. 2 presents the variation in effort distribution based on software criticality. The chart illustrates how critical software audits naturally demand a higher share of time and resources in the initial stages, driven by the need to ensure compliance with strict domain-specific requirements. This differentiation underlines the importance of tailoring the audit methodology according to the nature and significance of the audited software.

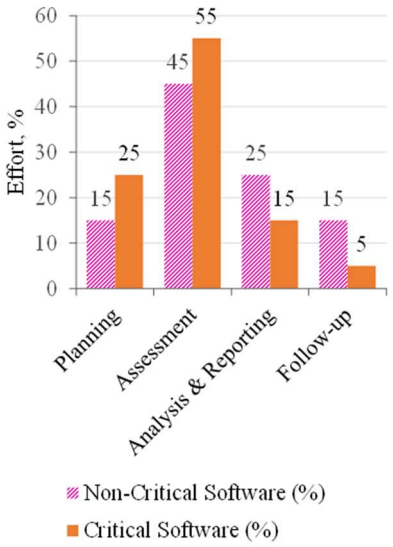

FIG. 2. Example of an image with acceptable resolution.

A. Mathematical Model of the Audit Methodology. To support practical implementation and resource planning, the audit effort may be expressed as a formal model. Eq. (1) defines the total audit effort as the sum of efforts across all audit stages and layers, weighted by importance and adjusted for software criticality. This formulation allows organizations to estimate and optimize audit resources, and to adapt the methodology for different software types. For example, audits of safety-critical systems typically involve higher weight coefficients and a criticality factor greater than one, reflecting the increased depth and rigor of assessment required:

$$E = \sum_{i=1}^{n}\sum_{j=1}^{m} W_{i,j} \cdot D_{i,j} \cdot C, \qquad (1)$$

where E is total audit effort, i is audit stage (planning, assessment, reporting, follow-up), j is audit layer (process, product, safety/security), $W_{i,j}$ is weight coefficient for stage and layer importance (based on experience or predefined), $D_{j,j}$ is duration or effort factor (estimated time/resource needs) for each stage/layer, C is criticality factor of the software (e.g., 1 for non-critical, >1 for critical software), n is total number of audit stages, m is total number of audit layers.

While Eq. (1) defines the overall audit effort as a function of planned activities, criticality, and resource allocation, it does not fully reflect the completeness and execution quality of the audit itself. Total effort can be high or low depending on audit scope and criticality, but it does not indicate whether al planned stages and layers of the audit were actually covered. This is particularly important in the context of complex energy sector software audits, where skipping certain activities or reducing audit depth may lead to incomplete or unreliable audit outcomes. To address this, it is essential to introduce a complementary metric that captures the coverage of the audit process, that is, the extent to which planned audit activities were actually performed. This led to the development of Eq. (2) is Audit Coverage Score, which quantifies the proportion of completed checks relative to the total planned checks across all audit layers and stages. Eq. (2) was constructed based on the logical structure of the audit methodology. Each audit consists of a predefined number of stages and layers (as defined in the audit model). For each combination of stage and layer, there is an expected audit check or activity. The model assumes that for every planned check, auditors can record whether it was fully completed (1), partially completed (optional, e.g. 0.5), or not performed (0). By summing all completed checks and dividing by the total number of planned checks, the Audit Coverage Score expresses the audit completeness as a ratio ranging from 0 to 1 (or 0% to 100%). This metric is particularly valuable for analysing audit quality across projects. For example, a low coverage score may indicate audit scope reduction due to time or resource constraints, while a high score reflects full execution of the planned audit methodology. Thus, Eq. (2) complements Eq. (1) by shifting focus from total effort to audit thoroughness, offering auditors and organizations an important additional indicator for assessing the reliability and integrity of audit results:

$$A_c = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} C_{i,j}}{n \times m}, \qquad (2)$$

where A_c is audit coverage score (0 to 1 or 0% to 100%), $C_{i,j}$ is conducted check (1 if completed, 0 if not) for stage i and layer j, n is number of audit stages, m is number of audit layers.

While Eq. (2) is Audit Coverage Score reflects how thoroughly the audit process has been executed, it does not provide any insight into the results or findings of the audit. An audit may achieve high coverage, meaning that all planned checks were conducted, but the audit findings themselves could still vary significantly from minimal issues to multiple critical nonconformities. In the context of auditing software engineering practices, particularly in critical domains such as the energy sector, understanding and quantifying the number and severity of issues detected is essential for evaluating both the current state of processes and the effectiveness of the audit itself. This is where the concept of a Nonconformity Rate, Eq. (3), becomes important. Eq. (3) was developed to quantify the proportion of detected nonconformities relative to the total number of checks performed during the audit. This approach is grounded in audit practice, where each check performed, as defined in Eq. (2), results in either a conforming outcome (no issue) or a nonconforming

outcome (finding). By dividing the total number of nonconformities by the number of conducted checks, the Nonconformity Rate provides a normalized value that can be used for comparisons across audits of different scopes and sizes. This metric is particularly valuable for several reasons. First, it allows organizations to benchmark audit results over time or across projects, identifying patterns or trends in process performance. Second, it highlights areas of concern: a higher nonconformity rate may indicate process weaknesses or gaps in implementation, while a lower rate suggests better process adherence and software quality. Finally, in combination with the Audit Coverage Score Eq. (2), the Nonconformity Rate offers a balanced view of audit effectiveness is linking audit execution with audit outcomes. Thus, Eq. (3) complements Eq. (2) by moving from the evaluation of audit process completeness to the assessment of audit findings, creating a more holistic and actionable model for software audit analysis:

$$N_R = \frac{N_F}{T_C}, \tag{3}$$

where N_R is nonconformity rate, N_F is number of nonconformities found, T_C is total number of checks conducted.

While Eq. (3) is the Nonconformity Rate provides valuable insights into the quantity of issues found during the audit, it does not offer detailed information about the quality and maturity of processes in each audit layer. In practice, not all nonconformities are of equal significance. Some may represent minor process deviations, while others indicate serious gaps in safety, security, or product quality management. Therefore, a more nuanced approach is required to assess not only the number of issues, but also the overall maturity and robustness of audited processes. To address this need, Eq. (4) is a Process Maturity Score has been developed. This formula introduces a way to quantify the maturity level of processes within each audit layer (Process, Product, Safety and Security). By evaluating individual process elements and assigning them scores based on their implementation status, auditors can calculate an average score that reflects how well-developed and stable the processes are:

$$M_L = \frac{\sum_{k=1}^{p} S_k}{p}, \tag{4}$$

where M_L is process maturity score for a particular audit layer (scale from 0 to 1 or 0% to 100%), S_k is score for each process element (e.g., 1 for fully implemented, 0.5 for partially implemented, 0 for not implemented), p is total number of process elements assessed in that layer.

This metric allows auditors to distinguish between layers with mature, well-established processes and those with significant room for improvement. For example, in audits of critical energy software, the Safety and Security Audit Layer is expected to achieve a high Process Maturity Score due to strict regulatory requirements (IEC 61508, IEC 62443). Conversely, in non-critical applications, lower scores might be acceptable for certain layers. By introducing Eq. (4), the audit methodology gains the ability to not only quantify effort Eq. (1), completeness Eq. (2), and issues Eq. (3), but also evaluate process strength and maturity, offering a comprehensive and balanced view of software engineering practices. As a result, Eq. (4) complements the existing model by closing the loop from effort and execution to outcome and process quality, and strengthens the methodological basis for systematic audit assessments and improvement planning.

B. Integration with quality management systems. The proposed mathematical model, consisting of audit effort estimation, coverage evaluation, nonconformity analysis, and process maturity assessment, offers a structured and quantifiable approach to software audits. However, for such a methodology to deliver lasting value, it must be integrated seamlessly into the organization's existing Quality Management System (QMS). Embedding the audit methodology within the QMS ensures that audit activities are aligned with organizational processes, contribute directly to continuous improvement, and support compliance with both internal policies and external regulatory requirements. Furthermore, integration facilitates audit repeatability, comparability across projects, and linkage to certification and regulatory inspections. The following section outlines how this methodology can be effectively integrated into QMS frameworks, with particular attention to the needs of the energy sector.

Effective software audits should not function as isolated or ad hoc activities. Instead, they should be embedded within the organization's broader Quality Management System (QMS) to ensure sustainability, traceability, and impact. The proposed audit methodology is designed to align naturally with the principles and processes defined in widely adopted QMS standards, such as ISO 9001, while also addressing specific regulatory requirements relevant to the energy sector [2].

Alignment with QMS Processes. The methodology's audit process stages (planning, assessment, reporting, and follow-up) map directly to the Plan-Do-Check-Act (PDCA) cycle that underpins most QMS frameworks. Audit planning supports the "Plan" phase by defining objectives and preparing criteria. Execution and assessment correspond to the "Do" and "Check" phases, where activities are performed and results evaluated. Finally, reporting and follow-up align with the "Act" phase, driving corrective actions and continuous improvement. By integrating audit activities into this cycle, organizations ensure that audit results are not only recorded but also used as drivers for process optimization and risk mitigation.

Support for Documentation and Records. The mathematical models proposed Eq. (1) to Eq. (4) produce quantitative results that can be systematically documented and stored in QMS records. Audit effort estimations, coverage ratios, nonconformity rates, and maturity scores provide objective evidence for management reviews, regulatory inspections, and certification audits. This data-driven approach enhances the transparency and accountability of software engineering audits.

Alignment with Energy Sector Regulatory Requirements. In the energy sector, regulatory bodies and

industry standards demand rigorous control and documentation of software development processes, particularly for safety- and security-critical systems. Standards such as IEC 61508 and IEC 62443 require demonstrable evidence of process execution, risk management, verification and validation, and issue resolution.

The proposed audit methodology directly supports compliance with these requirements. The Process Audit Layer ensures that safety and security processes are defined and followed. The Product Audit Layer verifies that resulting software products meet quality and reliability expectations. The Safety and Security Audit Layer focuses specifically on domain-specific risks and controls. Moreover, audit results are expressed in coverage, nonconformity, and maturity metrics can be mapped to regulatory reporting formats, making audits both practical and compliant. By integrating the audit methodology into the QMS, energy sector organizations can ensure that audits become a central element of their governance and assurance frameworks. This integration enhances readiness for regulatory inspections, strengthens internal controls, and fosters a culture of quality, reliability, and continuous improvement throughout the software lifecycle.

C. Software-Based Implementation of the Audit Methodology. To facilitate the practical adoption of the proposed audit methodology, a dedicated software system named SoftAssure was developed [22]. This platform serves as an interactive tool that supports audit planning, requirement analysis, validation tracking, and documentation within a unified digital environment. The system is implemented as a modular web application, allowing audit teams to manage multiple projects, enter findings, and evaluate compliance levels across software engineering processes. SoftAssure enables seamless integration of the mathematical models described in the methodology. For example, audit effort estimations and coverage metrics can be computed directly based on the data entered by users, while maturity assessments are derived from structured input regarding organizational processes. By combining data entry with automatic evaluation and visual feedback, the system empowers auditors to maintain consistency and transparency in assessment procedures.

The platform also supports linkage to broader Quality Management System (QMS) components. It provides interfaces for connecting audit items to risk entries, training records, and incident management logs, enabling traceability and process alignment. Furthermore, the system's structure allows organizations to embed audit checklists and findings into regular QMS documentation cycles, supporting evidence-based reporting and continuous improvement. Overall, SoftAssure not only demonstrates the feasibility of implementing the proposed methodology in practice, but also acts as a foundation for future research and extensions. Its modular design allows for domain-specific customizations, such as audit schemes for the energy or critical infrastructure sectors. The system was used to validate the audit process within educational and research environments and is freely available for demonstration and academic use [22].

V. CONCLUSION

The conducted research resulted in the development of a structured and domain-oriented methodological approach to auditing software engineering practices in the energy sector. The proposed methodology integrates general software lifecycle models with energy-specific regulatory requirements, offering a comprehensive framework for assessing process quality, product reliability, functional safety, and cybersecurity.

As a result of this research, a structured methodology has been developed and formalized, contributing to the advancement of knowledge in the field of software auditing for the energy sector. This methodology is grounded in three essential components, which together form an integrated and scientifically substantiated framework:

1. A multi-layered audit model, which defines Process, Product, and Safety and Security layers. This model ensures comprehensive evaluation and traceability of audit activities across the entire software lifecycle, addressing both general and domain-specific requirements;
2. A structured audit process model, which establishes Planning, Assessment, Reporting, and Follow-up stages. This process structure is aligned with recognized quality management principles, enabling integration into organizational QMS frameworks and supporting process improvement and compliance activities;
3. A mathematical formalization of the audit methodology, expressed through a set of quantitative models (Eq. 1 - Eq. 4). These models enable precise estimation of audit effort, measurement of coverage, analysis of audit findings, and evaluation of process maturity. Their introduction enhances the objectivity, comparability, and analytical rigor of software audits in industrial contexts.

Together, these methodological results represent a significant contribution to the field of software quality assurance and audit practices. They bridge the gap between general software engineering standards and the specific demands of the energy sector, providing auditors and organizations with a robust toolset for ensuring the reliability, safety, and security of software systems in critical infrastructure environments. Through detailed analysis of international standards the methodology ensures alignment with regulatory and best practice requirements relevant to critical energy software systems. The approach facilitates identification of weaknesses not only at the product level, but also within organizational processes, risk management procedures, and safety/security assurance activities.

The proposed mathematical models enhance audit objectivity and comparability, allowing auditors to present findings in a quantitative and structured manner. By applying these models, organizations can track audit completeness, identify recurring process weaknesses, and prioritize improvement actions based on nonconformity and maturity indicators.

The results of this research contribute to the scientific and practical advancement of software auditing methodologies, particularly for application in complex and highly regulated environments such as the energy industry. The developed framework supports not only internal

quality assurance efforts but also regulatory compliance, certification readiness, and continuous improvement initiatives.

Future work will focus on the practical validation of the methodology through industrial case studies and further refinement of the mathematical models, particularly in terms of weighting factors and criticality scaling, to enhance precision and applicability across diverse energy sector contexts.

AUTHOR CONTRIBUTIONS

Z.L. – conceptualization, investigation, writing-original draft preparation; I.L. – investigation, resources, writing-original draft preparation, supervision writing-review and editing.

COMPETING INTERESTS

The authors declare that they have no conflict of interest.

REFERENCES

[1] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on SCADA systems," in *Proc. Int. Conf. Internet Things, Cyber, Physical and Social Computing*, Dalian, China, 2011, pp. 380–388.

[2] ISO 9001:2015. *Quality management systems - Requirements*. Geneva, Switzerland: International Organization for Standardization, 2015.

[3] International Organization for Standardization and International Electrotechnical Commission, *ISO/IEC 12207: Systems and software engineering – Software life cycle processes*. Geneva, Switzerland: ISO/IEC, 2008.

[4] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[5] C. N. Amoo, B. Eckman, and J. R. New, "A multicriteria framework for assessing energy audit software for low-income households in the United States," *Energy Efficiency*, vol. 18, no. 1, p. 12, 2025.

[6] R. Abbas, et al., "Adopting Secure Software Development Practices to Improve Financial Transactions in the Banking Sector," unpublished.

[7] R. K. Jena, "Factors influencing blockchain adoption in accounting and auditing in the face of Industry 4.0: a multi-criteria decision-making approach," *J. Accounting & Organizational Change*, 2025.

[8] M. A. Tucker, *The Impacts of Software Development Methodologies on New Model Success Rates in the US Automotive Industry*, Ph.D. dissertation, Walden Univ., USA, 2025.

[9] A. Diyab, et al., "Engineered Prompts in ChatGPT for Educational Assessment in Software Engineering and Computer Science," *Education Sciences*, vol. 15, no. 2, p. 156, 2025.

[10] V. V. Ganapathy and S. Sampath, "Regulatory and Security Compliance for Software In Cloud Ecosystems–a Systematic Literature Review," unpublished.

[11] A. Sholihin and M. Salman, "OSCAT: A Comprehensive Tool for Automated CIS Benchmark Auditing," *Asian J. Eng., Social and Health*, vol. 4, no. 2, pp. 443–452, 2025.

[12] V. Terragni, et al., "The Future of AI-Driven Software Engineering," *ACM Trans. Softw. Eng. Methodol.*, 2025.

[13] International Organization for Standardization/International Electrotechnical Commission, *ISO/IEC 12207:2017 Systems and software engineering — Software life cycle processes*. Geneva, Switzerland: ISO/IEC, 2017.

[14] International Organization for Standardization/International Electrotechnical Commission, *ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes*. Geneva, Switzerland: ISO/IEC, 2015.

[15] IEEE Standards Association, *IEEE Std 730-2014 - IEEE Standard for Software Quality Assurance Plans*. New York, NY, USA: IEEE, 2014.

[16] IEEE Standards Association, *IEEE Std 1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation*. New York, NY, USA: IEEE, 2016.

[17] International Organization for Standardization/International Electrotechnical Commission, *ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. Geneva, Switzerland: ISO/IEC, 2011.

[18] International Electrotechnical Commission, *IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems*. Geneva, Switzerland: IEC, 2010.

[19] International Electrotechnical Commission, *IEC 62443 (multiple parts): Security for industrial automation and control systems*. Geneva, Switzerland: IEC, 2013–2021.

[20] International Organization for Standardization/International Electrotechnical Commission, *ISO/IEC 33001–33099:2015 Information technology — Process assessment*. Geneva, Switzerland: ISO/IEC, 2015.

[21] CMMI Institute, *CMMI for Development, Version 2.0*. Pittsburgh, PA, USA: CMMI Institute, 2018. [Online]. Available: https://cmmiinstitute.com/cmmi

[22] I. Liutak, *SoftAssure: A Web-based Tool for Software Audit Management*, 2025. [Online]. Available: https://github.com/iliutak/softassure

Ihor Liutak

Doctor of Technical Sciences, Professor at the Department of Software Engineering, Ivano-Frankivsk National Technical University of Oil and Gas. Specializes in component-based software engineering, auditing software processes in the energy sector, and data visualization. Research interests include approaches to software audits, development. Author of more than 100 scientific papers.

ORCID ID: 0000-0001-8960-5871

Zinoviy Liutak

PhD in Technical Sciences, Professor at the Department of Information and Measurement Technologies, Ivano-Frankivsk National Technical University of Oil and Gas. Specializes in standardization, verification and validation of software engineering processes, non-destructive systems of quality assurance, and software quality management. Author of more than 100 scientific papers.

ORCID ID: 0009-0000-8323-6980

Методологічний підхід аудиту розробки програмного забезпечення в енергетичному секторі

Ігор Лютак[1,*], Зіновій Лютак[2]

[1] Кафедра інженерії програмного забезпечення, Івано-Франківський національний технічний університет нафти і газу, Івано-Франківськ, Україна

[2] Кафедра інформаційно-вимірювальних технологій, Івано-Франківський національний технічний університет нафти і газу, Івано-Франківськ, Україна

*Автор-кореспондент (Електронна адреса: ihor.liutak@nung.edu.ua)

АНОТАЦІЯ Зростаюча складність програмних систем в енергетичному секторі, особливо тих, що забезпечують управління розподіленими та відновлюваними джерелами енергії, вимагає впровадження структурованих і орієнтованих на галузь методологій аудиту. Забезпечення надійності, безпеки та захищеності програмних продуктів у цьому контексті є критично важливим через посилення залежності промислової та енергетичної інфраструктури від автоматизованих і програмно-керованих рішень. У цій статті запропоновано комплексний методологічний підхід до аудиту практик розробки програмного забезпечення, адаптований до потреб енергетичного сектора. Розроблена методологія базується на інтегрованій моделі аудиту, яка визначає рівні процесів, продукту та функціональної безпеки і захищеності, що дозволяє здійснювати цілісну і систематичну оцінку. Додатково вона включає структурований процес аудиту, узгоджений з принципами систем управління якістю, що охоплює всі ключові етапи — від планування до завершального аналізу та рекомендацій. Важливою особливістю підходу є математична формалізація аудиторської діяльності, що включає моделі для оцінки зусиль, вимірювання повноти аудиту, аналізу невідповідностей та оцінки зрілості процесів. Ці моделі підвищують об'єктивність і аналітичну точність аудиту, дозволяючи організаціям кількісно порівнювати результати між проектами та циклами аудитів. Запропонована методологія розроблена на основі глибокого аналізу міжнародних стандартів, включаючи ISO/IEC 12207, ISO/IEC 25010, IEC 61508, IEC 62443 та ISO 9001, і покликана усунути розрив між загальними вимогами до програмної інженерії та галузевими потребами, пов'язаними з функціональною безпекою, кіберзахистом та експлуатаційною надійністю. Результати дослідження сприяють розвитку методів аудиту у сфері програмної інженерії та забезпечують науково обґрунтований і практично орієнтований інструмент для підвищення якості, безпеки та відповідності програмних систем, що використовуються в енергетичному секторі.

КЛЮЧОВІ СЛОВА аудит програмного забезпечення, енергетичний сектор, програмна інженерія, функціональна безпека, кібербезпека.