ISSN 2707-1898 (print)
Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2025.01.086
Article received 16.04.2025 p.
Article accepted 01.05.2025 p.
UDC 004.4

Correspondence author
M. Y. Lyashkevych
mariia.liashkevych@lnu.edu.ua

M. Y. Lyashkevych, V. Y. Lyashkevych, R. Y. Shuvar
Ivan Franko National University of Lviv, Lviv, Ukraine

SECURITY AND OTHER RISKS RELATED TO LLM-BASED SOFTWARE DEVELOPMENT

Generative Artificial Intelligence (GAI) is a new technology, and even though its capabilities are thoroughly being inspected and applied in different industries, it continues to develop intensively bringing in new types of risks in the software development domain. The variety of large language models (LLMs) emergency has led to changes at all stages of the Software Development Life Cycle (SDLC). Thus, the main objectives of the article are to identify and understand the potential risks associated with LLM-based software development and to identify the best approaches to mitigate the risks. The paper presents observations of common software architectures based on LLM, risks and their impact on traditional SDLCs, approaches to testing and software quality assessment, and an analysis of how LLM has changed the software development industry. The widespread LLM-based application architectures already have a traditional set of components like a chatbot channel for interaction with users, a knowledge base for providing the appropriate context to LLM, the components required for authentication and providing secure knowledge processing during the session and prompt accessibility, retrieval-augmented generation system (RAG) and evaluation modules. It is shown that integrating LLM into software generates unique risks that require changes to the already established SDLC at the level of architectural modifications, the evaluation system, and best practices for risk mitigation. Each component of the LLM-based system can generate specific risks in each SDLC stage. To more effectively identify and delineate risks, risk naming and its description were considered, and the traditional risk taxonomy was updated with an LLM-based software taxonomy. It is worth noting that another stage has been added to the conventional SDLC, which is related to the selection and management of personnel. This is because GAI is now a new technology and requires changes to the traditional composition of specialists. The risk examples, shown in the paper, are presented with risk identifiers that help identify the risk in a specific SDLC and connections to other related risks. It is an assumption that one risk has a minimum of 8 risk identifiers, what is equal to 8 SDLC stages. It explained how the risk manifests in different SDLC stages, which were defined by subject matter experts (SMEs) from IT companies. Finally, some sets and concepts were formalized for future calculations and research of the identified risks.

Keywords: software development life cycle, software risks, security risks, LLM-based software architecture.

Introduction

Relevance of research. Integrating LLMs into software development has brought a transformative era, offering powerful enhancements for natural language processing (NLP) and automation. However, this advancement requires a critical analysis of the associated risks and challenges considering key items: the imperative changes in the Software Development Life Cycle (SDLC) due to LLM trends; the architectural shifts prompted by LLM applications; the essential role of evaluation frameworks; the identification of risks within LLM-based SDLC and their mitigation frameworks; and additional pertinent considerations.

The object of research is the traditional taxonomy of software development risks.

The subject of research is the risks associated with the architectures of applications that incorporate LLMs.

The purpose of the research is to analyse LLM-based application architectures, aiming to identify the appropriate risk categories, their samples, and the initial formalization of risk attributes.

It becomes clear that the integration of LLM into software brings new challenges and risks; therefore, the problems related to new risk identification, as well as their mutual influence on other already known ones. To achieve this purpose, the following main research objectives are identified:

1) identify problem areas in the LLM-based application architecture where new risks are being generated;

2) analyse how new risks manifest in different SDLC stages of LLM-based applications overview;

3) clarify the definitions and standardize the namings for newly discovered risks;

4) analyse and update the traditional taxonomy with new risk categories related to LLM-based application architectures and mapping of the risk categories to SDLC stages;

5) define risk indicators for new risks and find out some samples of risks per each new category;

6) define the possible initial mathematical formalization of risks and steps to mitigation in the case of LLM-based application development.

Analysis of recent research and publications. Indeed, the emergence of LLM has significantly impacted the traditional SDLC, requiring adaptations to accommodate these models' unique characteristics. The common SDLC stages – planning, requirements analysis, design, implementation, testing, deployment, and maintenance – must evolve to address the complexities introduced by LLM.

A place of risks in widespread LLM-based application architectures. An LLM integration into software systems forces a rethink of traditional architectural models. LLMs, with their data-intensive and computationally demanding nature, create unique challenges and opportunities for system design. The articles [1, 2] have represented the most comprehensive observation from the fundamental concepts of the LLM traditional pipeline in the training phase, through dataset processing and the wide range of LLM-based applications solving real-world problems, to the exploration of LLM deployment open issues and challenges in real-world scenarios. In terms of applications, the paper [1] discusses the deployment of LLMs across various domains, including NLP tasks like translation, summarization, and question-answering tasks.

LLM is not being implemented into the application as an extra component, it requires some orchestrators that depend on application assignments. The open issues and challenges associated with LLMs, such as ethical concerns, biases in training data, computational resource demands, LLM frameworks and orchestrators, RAG architectures, and effective deployment of LLMs in real-world scenarios were inspected in [1-4]. Chatbot is one of the most widespread assignments for LLM-based applications. Thus, in [1-2], this question was considered and highlighted as prominent because GAI allows the preparation of intelligent, tailored assistance solving pretty complex questions. By integrating Generative Pre-trained Transformer-based models with Retrieval Augmented Generation (RAG) and Function Calling features, the described architecture in [2] aims to provide a co-pilot-like experience, guiding users through understanding requirements and generating configuration scripts. LLMs necessitate robust data pipelines to manage the continuous flow of training and inference data. Architectures must incorporate components for data collection, preprocessing, storage, and retrieval, ensuring data quality and consistency. Efficient data handling is critical to maintaining the performance and reliability of LLM applications [2–4].

Based on the architecture described in [1–4], we can assume that the following software development risks emerge due to the integration of LLMs as core orchestrated components rather than as simple add-ons. Thus, we want to highlight some categories of related risks such as architectural, integration, data pipeline, content quality, model-based, ethical and regulatory, deployment, operational, and human interaction (chatbot use case).

Software architectures are increasingly adopting modular designs to accommodate LLMs. This approach encapsulates LLM functionalities within distinct modules or microservices, promoting scalability and maintainability. Such modularization allows for independent development and deployment of LLM components, facilitating updates and experimentation without disrupting the entire system. In the works [5–8], agent-based software architecture was considered.

In [6], SALLMA, a software architecture for multi-agent LLM-based systems addressing single-agent limitations like lack of memory and task flexibility, was introduced. SALLMA is designed as a distributed, modular architecture to potentially support scalability, adaptability, and resilience. It uses an Operational and Knowledge Layer to manage tasks and context. A proof of concept demonstrates SALLMA's viability using modern AI and container technologies in real-world scenarios. This architecture is interesting for us because it represents a distributed system. Given the computational demands of LLMs, architectures must be designed for scalability. This involves leveraging distributed computing resources, load-balancing optimization, and implementing caching strategies. Cloud-based solutions are often employed to provide the necessary infrastructure, offering flexibility and scalability to meet varying workloads. Analyzing SALLMA [6] architecture, we found some risk categories such as agent coordination failures, memory and context drift, security and privacy, model bias and validation, system scalability and performance, integration complexity, maintainability, and regulatory compliance.

LLM applications require some security considerations because integrating LLMs introduces new security concerns, including vulnerabilities to adversarial attacks and data breaches. Architectures must incorporate robust security measures, such as input validation, output sanitization, and access controls, to safeguard against these threats. Implementation of the required techniques can generate security risks [8]. Additionally, secure APIs and endpoints are essential to protect data integrity and confidentiality.

Deploying LLM-based applications requires a comprehensive evaluation framework to assess their performance, reliability, and ethical implications. Traditional evaluation metrics are insufficient to capture the multifaceted nature of LLM, which drives the development of specialized application features. Evaluating LLM program performance requires metrics that, in addition to accuracy, include measures of fluency, coherence, and contextual

relevance. As usual, we can use benchmarking with standardized data sets and tasks to facilitate comparative analysis, helping to identify strengths and weaknesses, but sometimes we should create a golden dataset. Here, we recognized the next categories of possible risks such as security, privacy, evaluation, reliability, integration, and ethical and regulatory.

Risks in SDLC of LLM-based applications overview. Integrating LLM into the SDLC poses several risks that must be identified and managed to ensure the successful deployment of robust and ethical applications [9]. These risks span different stages of the SDLC, impacting the overall security, reliability, and ethical integrity of software systems.

There are a lot of different risk categories related to SDLC of LLM-based applications and their naming is not completely standardized. Therefore, created taxonomies [10–12] are good for starting the research. The paper [12] presents GUARD-D-LLM, a novel risk assessment engine for the downstream use of LLMs, which operates the risks that are being recognized and named by LLMs – it is one of the ways to standardize the risk taxonomy. In any case, it is a step ahead. Of course, it is very hard to inspect all possible related risk categories, thus, we would start from the most important ones based on some LLM-based application features such as data poisoning, rapid injection attacks, model theft, insecure output processing, denial of service (DoS), supply chain vulnerabilities, sensitive information disclosure, legal and regulatory compliance, overreliance on LLMs, SDLC stage-specifics, prompt engineering, human-agent collaboration, evaluation and testing of LLMs, tooling and infrastructure, ethical and societal, concept and knowledge drift, model updating, model versioning, costs and sustainability.

Data Poisoning – is one of the possible risk categories because LLMs are susceptible to data poisoning attacks, where attackers manipulate training data to negatively affect the behavior of the model. This can lead to the model producing biased or harmful results, undermining trust and reliability. Implementing rigorous data validation and traceability is essential to mitigate this risk [9, 13, 14]. Data poisoning involves the intentional manipulation of training data to introduce malicious behavior or biases into an LLM. Attackers can inject corrupted data into the training set, causing the model to learn and maintain unwanted patterns. As a result, LLM creates insecure code that spreads the bias or disseminates false information [13]. Implementing strict data validation, traceability, and the use of anomaly detection techniques is important to mitigate this risk.

Rapid injection attacks occur when attackers create data designed to manipulate the behavior of an LLM, leading to unauthorized actions or the disclosure of sensitive information. For example, an attacker could introduce a specially crafted prompt that causes the LLM to generate malicious code or disclose sensitive data. To protect against such attacks, it is essential to implement strict input validation, contextual filtering, and constant monitoring of model interactions [13, 15–17].

In the case of model theft [18], LLMs represent significant investments in intellectual property and computing technology. Unauthorized access or theft of model parameters can lead to competitive disadvantages and security breaches. Implementing robust access controls, encryption, and monitoring unauthorized access attempts is critical to protecting against model theft.

Insecure output processing refers to inadequate validation and sanitization of LLM-generated content before it is used or displayed. This can lead to the execution of malicious code, the disclosure of sensitive information, or the distribution of inappropriate content. Ensuring that all results are thoroughly vetted, implementing content filtering mechanisms, and maintaining human oversight are critical measures to address this risk [19].

Denial of Service (DoS) risk is related to attacks that aim to overload the LLM with incoming data that exhausts its computing resources, leading to reduced performance or complete unavailability. Such attacks can disrupt services and impact business operations. Implementing rate limiting, resource allocation controls, and monitoring for unusual activity patterns can help mitigate the risks of DoS [20].

Software development is based on complex software supply chains that offer efficiency and reuse but pose significant security risks. Attacks on a single link can impact entire systems, especially through compromised third-party components [21, 22]. LLM-based applications often rely on third-party components such as pre-trained models, datasets, unstructured data, and other libraries. Compromises in any part of this supply chain can create vulnerabilities in the system. Conducting thorough validation of external components, maintaining dependency inventories, and applying timely updates and patches are key methods for managing supply chain risk [23]. Paper [21] proposes a holistic end-to-end risk management (FARM) framework helping companies to identify, assess, respond, and monitor security threats in open-source and third-party software across the SDLC.

Large Language Models (LLMs) offer powerful capabilities for language understanding, generation, and security applications such as vulnerability detection and privacy protection. LLMs trained on large datasets may inadvertently remember and disclose confidential information in the training data. This can lead to privacy breaches and regulatory non-compliance. Using differentiated privacy and access control can help prevent unintended disclosure. However, their human-like mindset also allows them to be abused in offensive attacks. This study [24] recognizes their impact as “good,” “bad,” and “ugly,” highlighting both their cybersecurity potential and the need for deeper research into new threats and defenses.

The European Data Protection Board (EDPB) represented the updated document [25], which offers practical recommendations for developers, users, and decision-makers to manage privacy and data protection risks in

Large Language Model (LLM) systems. It supports General Data Protection Regulation (GDPR) compliance and complements Data Protection Impact Assessment (DPIA) requirements. The guide covers LLM fundamentals, data flows, privacy risks, risk assessment, mitigation strategies, residual risk assessment, and monitoring. It includes real-world examples and tools for systematic risk management. Developers receive guidance on integrating risk controls into system design, and users receive support in assessing risks before deployment. The methodology helps companies build secure, confidential LLM-based applications and make informed and responsible AI decisions because the usage of LLM must comply with various legal and regulatory requirements, including those relating to data protection, intellectual property, and the ethical use of AI [26]. Non-compliance can lead to legal sanctions and reputational damage. To manage legal and regulatory compliance risks, it is necessary to be aware of relevant regulations, conduct regular compliance audits, and implement a management system.

The article [26] examines the risks of exposing confidential data when integrating large language models (LLMs) into scientific workflows. It introduces "Data Shield", a framework designed to detect data leaks, summarize privacy policies, and visualize data flows, ensuring alignment with organizational policies. Ongoing user research is aimed at evaluating its effectiveness in real-world scenarios.

The overreliance on LLMs for critical decision-making processes can be risky, especially given their limitations and potential for error. Ensuring that LLM results are used to augment human judgment, not replace it, and maintaining mechanisms for human review and intervention are important to mitigate this risk [27, 28]. The paper [27] explores the ethical and educational impacts of relying on Large Language Models (LLMs) for critical thinking. It highlights risks to human agency, changes in educational frameworks, and the need to preserve reasoning skills. The paper [27] proposes models for human-LLM interaction and calls for ethical guidelines and interdisciplinary research on AI-augmented cognitive processes. In [28], software engineers have integrated LLMs like ChatGPT into workflows. With mixed results, they have revealed failures, causes, and mitigation strategies guiding future human-AI collaboration in software engineering.

Despite modern tools for building software projects and several risk management models, software still has high failure rates [29]. Improper risk assessment during software development was a major reason behind these unsuccessful projects as risk analysis was done on overall projects. The work [29] attempts to identify key risk factors and risk types for each of the development stages of SDLC, which would help to identify the risks at a much earlier stage of development. Empowering the application by GAI, the SDLC has evolved through agile methods, emphasizing speed and repeatability. Thus, we should map SDLC Stage-Specific Risk to software requirements because, with generative AI like ChatGPT, the software development process is poised for major transformation, potentially redefining the SDLC stages and significantly impacting the roles of software engineers [30].

The article [31] highlights security risks in prompt engineering during operational design, including instant implementation, leakage, jailbreaking, and manipulation by opposing parties. It addresses model poisoning, contextual drift, and social engineering. The focus is on safeguards such as input sanitary treatment, rapid isolation, and ethical constraints to ensure the integrity of the AI system, its resilience, and ethical operation across applications.

As is known, LLMs have given rise to promptware, a new paradigm where natural language prompts replace code. Unlike traditional software, promptware is ambiguous and non-deterministic, creating unique challenges. Paper [32] introduces promptware engineering – a structured methodology to systematically design, test, and evolve prompts – bridging software engineering principles with LLM-specific development needs and prompt engineering. LLMs shift software paradigms toward operational software, relying on natural language instead of code. This creates engineering and security risks, including rapid implementation, context drift, and ambiguous behavior due to the probabilistic nature of LLMs. Paper [31] proposes a systematic engineering framework for operational software, while [32] outlines critical threats and precautions during operational design. Together, they highlight the urgent need for structured development practices and robust security for LLM-based applications [31, 32].

Both papers [33, 34] discuss the critical role of integrating human oversight with AI systems to enhance cybersecurity and AI security. The [33] highlights the importance of combining AI's pattern recognition and threat detection capabilities with human contextual understanding leads to more effective risk management strategies but the [34] highlights the limitations of current AI assessments that lack human interaction components, proposing a framework for Human Interaction Evaluation (HIE) to assess the impact of AI systems on society. In our opinion, they both argue for a synergistic approach where human judgment complements AI's effectiveness in solving complex tasks, mitigating the human-AI interaction and oversight risks.

The papers [35–37] represented the results in energy efficiency, performance, and some metrics. These items are related to sustainable development and can generate some environmental and energy efficiency impacts, which are being recognized as appropriate risk categories, such as environmental impacts and energy efficiency. The [35, 37] highlight the importance of optimizing LLM to improve energy efficiency. Techniques such as quantization and local inference can significantly reduce energy consumption. The choice of model should take into account the complexity of the task and energy constraints; smaller models are suitable for simple tasks, while larger models, although more powerful, consume more energy. Hardware configurations also play a crucial role in energy efficiency.

Research results and their discussion

Naming clarification of "Risks in Software Development" taxonomy. The paper [10] represented the taxonomy for the "Risks in the Software Development Lifecycle" subject domain. It was 10 categories in the root. Analyzing the current state of the problem, using the results of scientific research in LLM-based applications and communicating with subject matter experts (SMEs) in the IT industry, we have updated the namings of those 10 risk categories as:

- Project Governance and Planning. These steps cover risks associated with poor project scope definition, unrealistic timelines, unclear objectives, insufficient stakeholder engagement, and impacts of AI-related uncertainty in project vision and management.
- Requirements Volatility and Ambiguity. Refers to unclear, contradictory, or constantly changing requirements, exacerbated by misinterpretation of LLM business needs when used during discovery.
- Architecture and System Design. Risks are associated with poor design decisions, failure to address non-functional requirements, or monolithic LLM integration without modularity. Includes design issues related to agent orchestration and context preservation.
- Implementation and Code Quality. The steps cover coding errors, unsafe practices, and poor technical support. Code generated by LLM may introduce syntactical correctness but semantic errors or energy inefficiency.
- Testing, Validation, and Verification. LLM-based systems require new testing strategies. Risks include inadequate assessment of speed, consistency, contextual alignment, lack of golden datasets, or non-determinism in results.
- Deployment and Release Management. The steps eliminate failures in CI/CD pipelines, configure the issues with containers and untested model behaviour in production, and include the risk of rapid release cycles without validation of model performance.
- Operational and Maintenance. Post-deployment risks include drift monitoring, unforeseen changes to source data after updates, and the inability to manage model versions or backward compatibility.
- Team Skills and Capability Gaps. Due to insufficient knowledge of AI tools, model lifecycle, operational design, and ethics. Risks from low LLM literacy or poor interdisciplinary collaboration.
- Stakeholder and Human Interaction. Includes the risk of over-trusting a human, insufficient analysis of LLM results, user trust issues, and failed handoffs between humans and LLM agents.
- External Legal, Regulatory, and Market. Risks related to GDPR compliance, intellectual property laws, export controls, and the changing AI regulatory landscape. Include risks related to changes in global AI policy and market.

Also, we have defined risk categories of LLM-based application development. Here we have found 20 crucially important categories:

- Training Data Poisoning. Malicious manipulation of training datasets to introduce biases or backdoors, leading to inappropriate behavior or harm to society. Includes issues with tracing the origin of training.
- Prompt Injection and Adversarial Inputs. Risks from crafted inputs alter the behavior of a model to leak data, create malicious content, or perform unauthorized actions. Includes jailbreak attacks.
- Model Theft and IP Leakage. Unauthorized access to LLM weights, configurations, or API keys. It exposes IP, increases cloning risks, and can lead to the leak of confidential information.
- Unsafe Output Generation. Risks associated with a lack of post-processing, verification, human-in-the-loop filtering, hallucinations, code injection, or malicious recommendations.
- Denial of Service and Resource Exhaustion. LLM endpoints are overloaded or GPU/CPU resources are exhausted, which can lead to performance degradation, service crashes, or be caused by complex prompts or scaling bottlenecks.
- Third-Party and Supply Chain Dependencies. Risks from compromised pre-trained models, datasets, or external libraries. Includes transitive risks in AI toolchains and open-source contributions.
- Confidential Information Exposure. Risks from inadvertent storage and retrieval of personally identifiable information, trade secrets or training on improperly prepared data.
- Compliance and Ethical Governance Gaps. Risk of violating privacy laws, ethical standards, or principles of fairness, including a lack of AI governance structures and accountability.
- Overdependence on Automation. Delegating important LLM decisions without checks. Causes risks to reliability, accountability, and erosion of human skills.
- SDLC Stage-Specific Risk Alignment. Unique and specific risks for each stage of the SDLC encourage staged risk mitigation planning, service deviations, data leakage during training, etc.
- Prompt Engineering Risks. Risks from poorly designed hints that mislead models, degrade inference quality or reinforce malicious patterns. Includes security-sensitive hint generation errors.
- Evaluation and Benchmarking Challenges. The risks of deploying poorly validated models due to weak or unrepresentative metrics and a lack of industry standards for assessing LLM performance.
- Toolchain and Infrastructure Risks. Configuration errors, version mismatches, or orchestration issues with containers, GPUs, and API limitations. LLMs often require specialized, evolving information.
- Bias, Toxicity, and Societal Harm. LLMs can reproduce and spread harmful stereotypes, misinformation, or cultural biases, causing reputetional or societal harm.

- Semantic Drift and Obsolescence. Risks associated with LLMs producing outdated or irrelevant outcomes due to the decay or inconsistency of time-sensitive knowledge.
- Model Update and Retraining Risks. Includes drift during fine-tuning, version confusion, and inadequate regression testing after updates.
- Versioning and Compatibility Management. Difficulty managing the compatibility of LLM versions and configurations, APIs, subsequent applications, etc.
- Cost and Computational Resource Constraints. High financial and environmental costs for inference, training, and fine-tuning, including excessive usage of GPUs in the cloud.
- Environmental and Sustainability Concerns. Risks due to high carbon footprint and energy consumption during training and inference that conflict with ESG (Environmental, Social, and Governance) objectives.
- Innovation and Technology Obsolescence. The rapid evolution of LLMs and frameworks can make solutions obsolete, increasing the costs of updating, retraining, and revalidation.

The proposed risk category names are close to terminology in the IT industry, which allows extending the taxonomy easily with extra appeared categories. Avoiding the ambiguity in the meaning of the proposed categories, including risks of LLM-based application development and combining traditional SDLC risk categories with new AI-related issues, we suggest the appropriate description for further recognition.

Mapping of risk categories to SDLC stages. The paper [10] represented 7 stages of SDLC, but, after analysis of LLM-based application architectures and development processes, we added an extra stage related to staffing, named Team & Management, in the Table 1 below. This depends on the fact that LLM-based architecture starts with scientific data analysis, unlike traditional software, for which the technical solution remains more important.

By the way, the traditional SDLC allows us to work efficiently with LLM-based applications, but with some changes. The traditional and LLM-based risk categories mapping on SDLC stages is shown in Table 1.

Table 1. Risk categories mapping to SDLC stages

No.	SDLC stages	Risk categories
1	Project Initiation and Planning	Project governance and planning risks, external legal, regulatory, and market risks, innovation and technology obsolescence
2	Requirement Analysis	Requirements volatility and ambiguity, compliance and ethical governance gaps, training data poisoning, prompt injection and adversarial inputs, bias, toxicity, and societal harm
3	Architectural Design	Architecture and system design risks, prompt engineering risks, toolchain and infrastructure risks, semantic drift and obsolescence
4	Development	Implementation and code quality risks, model theft and ip leakage, unsafe output generation, third-party and supply chain dependencies, confidential information exposure, overdependence on automation
5	Testing	Testing, validation, and verification risks, evaluation and benchmarking challenges, SDLC stages-specific risk alignment
6	Deployment	Deployment and release management risks, denial of service and resource exhaustion, versioning and compatibility management, cost and computational resource constraints
7	Maintenance and Operations	Operational and maintenance risks, model update and retraining risks, environmental and sustainability concerns
8	Team & Management	Team skills and capability gaps, stakeholder and human interaction risks

Examples of risks and their indicators. Risk definition plays an extremely important role, as it is a way to distinguish risks from each other. Identifying risks, we use the following rules for risk definition: risk name should consist of 2 to 10 words extending its context; we detect two possible types of relationships ("Related to" and "Depends on") with other risks; we define a mitigation plan (look at Table 2).

Table 2. Examples of risks for the category "Project governance and planning risks"

No.	Risks	Risk description	Related to	Depends on	Mitigation plan
1	2	3	4	5	6
1	Inadequate risk planning	Failure to identify and plan for risks early can lead to delayed mitigation and project failure	Stakeholder and Human Interaction Risks	Team Skills and Capability Gaps	Perform a comprehensive risk assessment at the beginning of the project
2	Misaligned goals	Different goals of stakeholders and developers can lead to project inefficiency or failure	Requirements volatility and ambiguity	Stakeholder and human interaction risks	Use stakeholder interviews and alignment workshops
3	Unrealistic timelines	Deadlines that do not reflect actual complexity increase the likelihood of rushed and poor-quality delivery	Implementation and code quality risks	Team skills and capability gaps	Use historical project data to inform planning and provide contingency reserves

Continuation of **Table 2**

1	2	3	4	5	6
4	Budget under-estimation	Underestimating the budget can cause resource shortages and disruptions	Cost and computational resource constraints	External legal, regulatory, and market risks	Apply bottom-up cost estimation and periodic budget reviews
5	Inadequate governance structure	Without a clear governance structure, decision-making and accountability suffer	Team skills and capability gaps	Stakeholder and human interaction risks	Establish project charters, roles, and escalation paths

Also, we have a definition of risk identifier [10] which consists of 7 to 20 words. This is the manifestation of the risk in a specific stage of the SDLC, i. e., one risk has a minimum of 8 identifiers corresponding to the number of stages in the SDLC. Sometimes, one risk has some indicators, and one risk indicator can point to some risks.

Chosen sets and formulas for risk attribute formalization. Risk assessment in LLM-based applications involves a systematic approach to identifying, evaluating, and mitigating potential vulnerabilities. The following steps describe a comprehensive risk assessment methodology:

- *Risk identification*. Start by cataloguing all potential risks associated with using LLM. This includes technical risks (e. g., data corruption, instant implementation), operational risks (e. g., model performance degradation), and compliance risks (e. g., data privacy breaches). Involving stakeholders and SMEs from different fields can provide a holistic view of possible risk factors.
- *Risk analysis*. Risk potential impact analysis and probability of occurrence involve assessing the severity of the consequences if the risk materializes and estimating the probability based on historical data, expert judgment, or statistical models.
- *Risk assessment*. Prioritizing risks helps to focus resources on addressing the most critical vulnerabilities. Creating a risk matrix can help to visualize and rank risks, facilitating informed decision-making.
- *Risk mitigation planning*. Developing and implementing strategies for risk mitigation may include technical measures (e. g., improving input validation to prevent rapid adoption), process improvements (e. g., regularly retraining a model to eliminate data drift), and policy interventions (e. g., creating a data governance infrastructure).
- *Monitoring and review*. Risk management is an ongoing process that continuously monitors the effecttiveness of mitigation strategies and the emergence of new risks. Regularly review and update the risk assessment to adapt to new threats and changes in the work environment.
- *Documentation and reporting*. Maintaining comprehensive documentation of the risk assessment process is essential. This includes detailed reports on identified risks, analysis methodologies, mitigation strategies, and ongoing monitoring results. Well-documented risk assessments aid in compliance reviews, knowledge sharing, and iterative improvement of risk management strategies.

Beyond traditional risks, LLM-based applications introduce additional factors that require attention:

- *Explainability and interpretability*. LLMs are often considered black box systems, meaning their decision-making process is not transparent. This creates problems with error correction, integrity, and compliance. Using explanation techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations), we interpret the model behavior.
- *Ethical and societal implications*. LLMs may unintentionally generate misinformation, biased results, or harmful content. Companies deploying LLM-based solutions should be aware of ethical considerations, such as fairness in AI (preventing bias in training data); content moderation (ensuring the safety and relevance of the results produced); and transparency (clearly stating when AI is used in decision-making).
- *Continuous monitoring and model management*. Unlike traditional software, LLMs degrade over time as new data trends emerge. Continuous monitoring is critical to identify performance deviations because it allows drift detection and periodic retraining models as requested.
- *Cost and computational efficiency*. LLMs require significant computational resources, making them expensive to train and deploy. Companies must balance model performance with infrastructure costs. Optimization of LLM usage through knowledge distillation, quantization, and model reduction allows for to reduction of resource consumption.

LLM-based software development fundamentally changes our approach to SDLC, architecture, and risk management. By implementing structured assessment systems, robust security measures, and ongoing monitoring, developers can harness the power of LLM while mitigating the associated risks.

All defined risks are represented in the taxonomy of risks. Thus, in choosing the best approach for risk calculations, we should define structured sets and hierarchies. Assuming we have the sets:

- SDLC stages:

$$\mathbf{S} = \{s_1, s_2, \ldots, s_8\}, \quad (1)$$

where S – is a set of 8 stages.

- Software components per stage:

$$\mathbf{C} = \{c_1, c_2, \ldots, c_i\}, \quad (2)$$

where C – is a set of i components.

- System requirements:

$$\mathbf{Q} = \{q_1, q_2, \ldots, q_l\}, \quad (3)$$

where Q – is a set of l requirements.

Risks:

$$R = \{r_1, r_2, \ldots, r_m\}, \quad (4)$$

where R – is a set of m risks.

- A dependency graph:

$$D(r_i, r_j) \in [0,1], \quad (5)$$

where D – is a set of dependencies between r_i risk and r_j.

- Traceability Mapping:

$$Trace \to 2^C, \quad (6)$$

where $Trace$ connects requirements to components.

Assume, that each risk has some attributes as shown in Table 3.

Table 3. Examples of risks attribute to formalisation

No.	Attribute	Formalisation	Description
1	Probability	$V(c,r) \in [0,1]$	The likelihood that risk r affects component c
2	Impact	$I(c,r) \in \mathbb{R}^+$	Severity of impact of risk r on component c
3	Exposure	$RE(c,r) = V(c,r) \times I(c,r)$	The risk exposure of r on c
4	Costs of mitigation	$C(r) \in \mathbb{R}^+$	Estimated effort / cost to mitigate risk r
5	Risk source	$O(r) \in S \cup Q \cup C$	Origin (phase or requirement) of risk

The risk aggregation formulas are shown below:

- Component-level exposure:

$$RE(c,r) = V(c,r) \times I(c,r). \quad (7)$$

- Stage-level exposure:

$$RE(s_i) = \sum_{c \in C(s_i)} \sum_{r \in R} E(c,r). \quad (8)$$

- Requirement-level exposure:

$$RE(q_j) = \sum_{c \in \mathrm{Trace}(q_j)} \sum_{r \in R} E(c,r). \quad (9)$$

- System-level total risk:

$$RE(q_j) = \sum_{c \in \mathrm{Trace}(q_j)} \sum_{r \in R} E(c,r). \quad (10)$$

- Propagate impact through graph:

$$I_{\text{eff}}(r_i) = I(r_i) + \sum_{r_j \in R} D(r_i, r_j) \times I(r_j). \quad (11)$$

- Contextual sensitivity risks based on domain context:

$$W_{\text{context}}(r) \in [0,2],$$

$$RE_{\text{adj}}(c,r) = W_{\text{context}}(r) \times V(c,r) \times I(c,r). \quad (12)$$

- Explainability index:

$$E(c,r) \in [0,1]. \quad (13)$$

Updated (10) system-level total risk:

$$RE_{\text{total}}(t) = \sum_{s_i \in S} \sum_{c \in C(s_i)} \sum_{r \in R} \left[W_{\text{context}}(r) \times D_{\text{eff}}(r) \times V(c,r,t) \times I(c,r,t) \times (1 - E(c,r)) \right], \quad (14)$$

where $RE_{\text{total}}(t)$ – is a system-level total risk at moment t; $W_{\text{context}}(r)$ – contextual weight; $D_{\text{eff}}(r)$ – risk dependency multiplier, which accounts for cascading risk; $(1 - E(c,r))$ – explainability penalty, which increases risk if the issue is poorly explainable; $V(c,r,t) \times I(c,r,t)$ – time-varying factors, which track risk evolution over time.

The proposed aforementioned formulas for risk assessment define a structured quantitative methodology for risk assessment in LLM-based software systems. They map risks to components, requirements, and SDLC stages, calculating risk by combining probability and impact. Risk propagation through dependency graphs models cascading effects. The final system-level risk score combines all adjusted component-level risks, weighted by dependency, contextual sensitivity, explainability, and time-varying risky behavior. These formulas are not stable yet; they just show our approach to risk calculations and will be updated during thorough testing.

Discussion of research results. Integrating LLMs requires a deep understanding of their capabilities and limitations. Stakeholders should define clear objectives considering factors such as data privacy, ethical implications, and potential biases inherent in LLMs. The planning and requirements analysis stages demand a multidisciplinary approach, involving data scientists, ethicists, and domain experts to ensure comprehensive requirement gathering.

The implementation stage involves coding and fine-tuning LLM, which requires extra expertise and access to significant computing resources due to LLM computing capacity. Developers must consider model selection techniques, data channels, integration points, and frameworks for computing optimization.

Traditional testing methodologies don't fit the testing requirements for LLM-based applications because, in addition to functional testing, emphasis should be placed on information security, assessing model accuracy, robustness to competing inputs, and ethical considerations such as bias detection. Automated testing systems should be comple-

mented with tools capable of assessing LLM-specific attributes with predefined metrics.

Deploying LLM-based applications requires continuous monitoring to detect model drift, performance degradation, and new biases. Maintenance involves regularly updating the training data and model parameters, ensuring the application is up-to-date and reliable.

The relationships between software requirements and software development risks were established using predefined risk indicators [10]. The proposed risk indicators at each stage of the SDLC and their connections with the selected software development methodology make it possible to better structure concepts in the taxonomy of software development risks. The proposed taxonomy of SDLC risks in [10] didn't consider the specifics of LLM integration in software architecture.

The scientific novelty of the obtained research results includes the LLM-based application architecture risk categories based on the investigation of impacts and changes in the SDLC stages, mapping of the revealed risk categories on SDLC stages based on SME assessment, updated traditional taxonomy of "Risks in Software Development" with revealed risk categories and their real samples, the approach for risk identification process by risk identifiers and risk attributes. The risk identifiers allow us to recognize risk manifestations at each SDLC stage, and risk attributes allow the risk calculation.

The practical significance of the research results consists of the ability to automate monitoring of the software risks at each stage of SDLC, including LLM-based application architectures, with the recognized risk manifestations per each stage of SDLC, and evaluate the software risks by simple risk attributes calculations. Also, the updated taxonomy allows for allocating or finding out the possible new software risks in the future.

Conclusions / Висновки

The integration of LLMs impacts SDLC, generating new categories of risks. The results of the research sketch an updated taxonomy of risks in the SDLC, focusing on both traditional software development and LLM-based applications. It revises the naming of the ten main risk categories and introduces twenty additional risks specific to LLM. These updated categories cover 8 stages of the SDLC, addressing new risks associated with LLM-based applications. The proposed mapping of risk categories across SDLC stages relies on the unique requirements of LLM-based architectures. This mapping is innovative and highlights specific risks at each stage using appropriate risk indicators.

The proposed risk assessment methodology uses a comprehensive set of formulas to calculate and aggregate risks, including elements such as probability, impact, risk, and mitigation costs. The methodology includes context-specific adjustments for cascading effects and explainability penalties, providing a quantitative basis for assessing risks throughout the SDLC. Especially for LLM-based applications, the paper emphasizes additional issues such as explainability, societal implications, context, and continuous monitoring.

References

[1] Raiaan, M., Mukta, S., Fatema, K., Fahad, N., Sakib, S., Mim, M. M. J., Ahmad, J., Ali, M. E., Azam, S. (2024). A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access,* 1–1. DOI: https://doi.org/10.1109/ACCESS.2024. 3365742.

[2] D'Urso, S., Martini, B., Sciarrone, F. (2024). A Novel LLM Architecture for Intelligent System Configuration. 28th International Conference Information Visualisation (IV), Coimbra, Portugal, 326–331. DOI: https://doi.org/10.1109/IV64223. 2024.00063.

[3] Jeong, C. (2023). A study on the implementation of generative AI services using an enterprise data-based LLM application architecture. arXiv preprint arXiv:2309.01105. DOI: https://doi.org/10.48550/arXiv.2309.01105

[4] Mailach, A., Simon, S., Dorn, J., & Siegmund, N. (2024). Practitioners' Discussions on Building LLM-based Applications for Production. arXiv preprint arXiv:2411.08574. URL: https://dblp.org/rec/journals/corr/abs-2411-08574.html

[5] Arslan, A. (2024). Exploring LLM-based Agents: An Architectural Overview. Current Trends in Computer Sciences & Applications: Lupine Publishers. DOI: http://dx.doi.org/10.32474/CTCSA.2024.03.000162

[6] Marco, B., Verdecchia, R., Vicario, E. (2025). SALLMA: A Software Architecture for LLM-Based Multi-Agent Systems, The 2nd International Workshop New Trends in Software Architecture (SATrends2025). URL: https://robertoverdecchia.github.io/papers/SATrends_2025.pdf

[7] Stieler, D., Schwinn, T., Leder, S., Maierhofer, M., Kannenberg, F., Menges, A. (2022). Agent-based modelling and simulation in architecture. *Automation in Construction*, Vol. 141. DOI: https://doi.org/10.1016/j.autcon.2022.104426.

[8] Carnì, D., Grimaldi, D., Nigro, L., Sciammarella, P. F., Cicirelli, F. (2017). Agent-based software architecture for distributed measurement systems and cyber-physical systems design. IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin, Italy, 1–6. DOI: https://doi.org/10.1109/I2MTC.2017.7969977.

[9] Secure Development for LLM Applications: Best Practices & Trends. Securityium. URL: https://www.securityium.com/secure-development-for-llm-applications-best-practices-trends/?utm_source=chatgpt.com

[10] Lyashkevych, M., Rohatskiy, I., Lyashkevych, V., Shuvar, R. (2024). Software risk taxonomy creation based on the comprehensive development process. *Electronics and information technologies*, 59–71. DOI: https://doi.org/10.30970/eli.27.5.

[11] Cui, T., Wang, Y., Fu, C., Xiao, Y., Li, S., Deng, X., Li, Q, et al. (2024). Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. arXiv preprint arXiv:2401.05778.

[12] Sundaraparipurnan, N., Sandeep, V. (2024). GUARD-D-LLM: An LLM-Based Risk Assessment Engine for the Downstream uses of LLMs. DOI: https://doi.org/10.48550/arXiv. 2406. 11851.

[13] Threat Modelling and Risk Analysis for Large Language Model (LLM)-Powered Applications. URL: https://ar5iv.labs.arxiv.org/html/2406.11007

[14] TIPS #5: LLMs in software development: the rewards are clear, but what about the risks? URL: https:// forge-pointcap.com/ perspectives/tips-5-llms-in-software-

development-the-rewards-are-clear-but-what-about-the-risks/?utm_source=chatgpt.com
[15] Can We Trust Large Language Models Generated Code? A Framework for In-Context Learning, Security Patterns, and Code Evaluations Across Diverse LLMs. URL: https://arxiv.org/abs/2406.12513?utm_source=chatgpt.com
[16] The Security Risks of Using LLMs in Enterprise Applications. URL: https://coralogix.com/ai-blog/the-security-risks-of-using-llms-in-enterprise-applications/?utm_source= chatgpt. com
[17] DeepSeek's Safety Guardrails Failed Every Test Researchers Threw at Its AI Chatbot. URL: https://www.wired.com/story/deepseeks-ai-jailbreak-prompt-injection-attacks/?utm_source=chatgpt.com
[18] LLM Security: Top 10 Risks and 7 Security Best Practices. URL: https://www.exabeam.com/explainers/ai-cyber-security/llm-security-top-10-risks-and-7-security-best-practices/?utm_source=chatgpt.com
[19] LLM Security: Top 10 Risks, Impact, and Defensive Measures. URL: https://www.acorn.io/resources/learning-center/llm-security/?utm_source=chatgpt.com
[20] Unveiling the Top 10 LLM Security Risks: Real Examples and Effective Solutions. URL: https://flyaps.com/blog/ unveiling-the-top-10-llm-security-risks-real-examples-and-effective-solutions/?utm_source=chatgpt.com
[21] Onu, K. (2021). Software Supply Chain Risk Management Framework. DOI: https://doi.org/10.13140/ RG.2.2. 36364. 94083.
[22] Badis, H., Sherali, Z. (2023). Software Supply-Chain Security: Issues and Countermeasures. Computer. 56. DOI: https:// doi.org/10.1109/MC.2023.3273491.
[23] Virendra, A., Soeren, F., Abdallah, D. (2024). LLM-based Vulnerability Sourcing from Unstructured Data, 634–641. DOI: https://doi.org/10.1109/EuroSPW61312.2024.00077.
[24] Yao Y., Duan J., Xu K., Cai Y., Sun Z., Zhang Y. (2024). A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. High-Confidence Computing, Vol. 4, I. 2. DOI: https://doi.org/ 10.1016/ j.hcc.2024.100211.
[25] Barbera, I. (2025) AI Privacy Risks & Mitigations – Large Language Models (LLMs). EDPB. URL: https:// www.edpb.europa.eu/system/files/2025-04/ai-privacy-risks-and-mitigations-in-llms.pdf
[26] Shanmugarasa, Y., Pan, S., Ding, M., Zhao, D., & Rakotoarivelo, T. (2025). Privacy Meets Explainability: Managing Confidential Data and Transparency Policies in LLM-Empowered Science. arXiv preprint arXiv:2504.09961.
[27] Duenas, T., Ruiz, D. (2024). The Risks Of Human Overreliance On Large Language Models For Critical Thinking. DOI: https://doi.org/10.13140/RG.2.2.26002.06082.
[28] Tie, J., Bingsheng, Y., Tianshi, L., Syed, A., Dakuo, W., Shurui, Z. (2024). LLMs are Imperfect, Then What? An Empirical Study on LLM Failures in Software Engineering. DOI: https://doi.org/10.48550/arXiv.2411.09916.
[29] Bibhash, R., Ranjan, D. (2015). A Study on Software Risk Management Strategies and Mapping with SDLC. DOI: https://doi.org/10.1007/978-81-322-2653-6_9.
[30] Pothukuchi, Ameya Shastri & Kota, Lakshmi Vasuda & Mallikarjunaradhya, Vinay (2023). Impact of generative AI on the software development life cycle (SDLC). 11
[31] Geroimenko, V. (2025). Key Security Risks in Prompt Engineering. DOI: https://doi.org/10.1007/978-3-031-86206-9_5.
[32] Chen, Zhenpeng & Wang, Chong & Sun, Weisong & Yang, Guang & Liu, Xuanzhe & Zhang, Jie & Liu, Yang (2025). Promptware Engineering: Software Engineering for LLM Prompt Development. DOI: https://doi.org/ 10.48550/ arXiv.2503.02400.
[33] Miracle, Agboola & Hoover, Rose (2024). Combining AI Systems and Human Oversight in Cybersecurity Risk Management AUTHOR. Cybersecurity and Law. 6. 9-15. URL: https://www.researchgate.net/publication/387322788_Combining_AI_Systems_and_Human_Oversight_in_ Cybersecurity_Risk_Management_AUTHOR
[34] Ibrahim, L., Huang, S., Ahmad, L., & Anderljung, M. (2024). Beyond static AI evaluations: advancing human interaction evaluations for LLM harms and risks. arXiv preprint arXiv: 2405.10632. DOI: https://doi.org/10.48550/ arXiv.2405. 10632.
[35] Khan, T., Motie, S., Kocak, S. A., & Raza, S. (2025). Optimizing Large Language Models: Metrics, Energy Efficiency, and Case Study Insights. arXiv preprint arXiv:2504.06307. URL: https://chatpaper.com/chatpaper/fr/paper/128038
[36] Solovyeva, L., Weidmann, S., Castor, F. (2025). AI-Powered, But Power-Hungry? Energy Efficiency of LLM-Generated Code. DOI: https://doi.org/10.48550/arXiv.2502.02412.
[37] Maliakel, P., Ilager, S., Brandic, I. (2025). Investigating Energy Efficiency and Performance Trade-offs in LLM Inference Across Tasks and DVFS Settings. DOI: https://doi.org/ 10.48550/arXiv.2501.08219

М. Ю. Ляшкевич, В. Я. Ляшкевич, Р. Я. Шувар

Львівський національний університет імені Івана Франка, Львів, Україна

БЕЗПЕКА ТА ІНШІ РИЗИКИ, ПОВ'ЯЗАНІ З РОЗРОБЛЕННЯМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Генеративний штучний інтелект (GAI) – це нова технологія, і навіть незважаючи на те, що її можливості ретельно перевіряють та застосовують у різних галузях, вона продовжує інтенсивно розвиватися, створюючи нові типи ризиків у сфері розроблення програмного забезпечення. Різноманітність великих мовних моделей (LLM) привела до змін на всіх етапах життєвого циклу розроблення програмного забезпечення (SDLC). Основні цілі статті – визначення та розуміння потенційних ризиків, пов'язаних із розробленням програмного забезпечення на базі LLM, і виявлення найкращих підходів для пом'якшення таких ризиків. Здійснено спостереження за ризиками та їхнім впливом на традиційні SDLC, подано загальні архітектури програмного забезпечення на основі LLM, підходи до тестування та оцінювання якості програмного забезпечення, аналіз того, як LLM змінили сферу

діяльності з розроблення програмного забезпечення. Показано, що інтеграція LLM у програмне забезпечення породжує унікальні ризики, які потребують змін в уже установленому SDLC на рівні архітектурних модифікацій, системи оцінювання та найкращих методів пом'якшення ризиків. З метою ефективнішого виявлення ризиків та їх розмежування розглянуто питання найменування й описання ризиків, оновлено традиційну таксономію ризиків за рахунок таксономії ризиків програмного забезпечення на основі LLM. Підкреслимо, що додано ще одну стадію до традиційного SDLC, яка пов'язана із підбором та керуванням персоналом. Це зумовлено тим, що сьогодні це нова технологія і потребує змін у традиційному складі спеціалістів. Наведені в роботі приклади ризиків подано з ідентифікаторами ризиків, які допомагають ідентифікувати ризик у конкретній SDLC, зв'язки з іншими пов'язаними ризиками. Формалізовано деякі множини та поняття для майбутніх обчислень та досліджень виявлених ризиків.

Ключові слова: життєвий цикл розробки програмного забезпечення, ризики програмного забезпечення, архітектура програмного забезпечення на основі великих мовних моделей, ризики безпеки.

Інформація про авторів:

Ляшкевич Марія Юріївна, асистент, кафедра системного проєктування. **Email:** mariia.liashkevych@lnu.edu.ua; https://orcid.org/0000-0002-9655-036X

Ляшкевич Василь Яремович, канд. техн. наук, доцент, кафедра системного проєктування. **Email:** vasyl.liashkevych@lnu.edu.ua; https://orcid.org/0000-0003-2810-6061

Шувар Роман Ярославович, канд. фіз.-мат. наук, доцент, завідувач кафедри системного проєктування. **Email:** roman.shuvar@lnu.edu.ua; https://orcid.org/0000-0001-6768-4695

Цитування за ДСТУ: Ляшкевич М. Ю., Ляшкевич В. Я., Шувар Р. Я. Безпека та інші ризики, пов'язані з розробкою програмного забезпечення на основі великих мовних моделей. *Український журнал інформаційних технологій*. 2025, т. 7, № 1. С. 86–96.

Citation APA: Lyashkevych, M. Y., Lyashkevych, V. Y., & Shuvar, R. Y. (2025). Security and other risks related to LLM-based software development. *Ukrainian Journal of Information Technology*, 7(1), 86–96. https://doi.org/10.23939/ujit2025.01.086