ISSN 2224-087X. Електроніка та інформаційні технології. 2024. Випуск 27. С. 59–71
Electronics and information technologies. 2024. Issue 27. P. 59–71

УДК 004.4 DOI: https://doi.org/10.30970/eli.27.5

SOFTWARE RISK TAXONOMY CREATION BASED ON THE COMPREHENSIVE DEVELOPMENT PROCESSES

M. Lyashkevych, I. Rohatskyi, V. Lyashkevych, R. Shuvar

System Design Department,
Ivan Franko National University of Lviv,
50 Drahomanova St., 79005 Lviv, Ukraine
mariia.liashkevych@lnu.edu.ua

Software risks are always a crucially important topic for research because the software development process is quite expensive. The competition is high enough to ignore it. Although the "golden" era for startup projects is slowly ending, the latest achievements in generative AI show that now is the time to "take risks" and capture the software market using this technology. Therefore, it is necessary to analyse already known risks and identify new risks associated with business models and market conditions with generative AI capacity.

The article analyses the already existing taxonomies of software risks, their advantages and disadvantages, the software development life cycle stages, and risk management activities in the conditions of different software development models. Using the proposed taxonomy, the noticed activities and processes are linked in one taxonomy, which allows easy identification of risks based on known software requirements and vice versa.

The created taxonomy has been validated by some subject domain experts who work at big IT companies. ChatGPT4 is one of the experts counting on the LLM capability to resolve the summarisation and text classification tasks. The practical results of the risk taxonomy are crucially important because we avoid LLM hallucinations and enable a taxonomy-driven approach to prompt engineering for risk management.

Keywords: software development risks, risk taxonomy, risk recognition, risk detection, taxonomy, software requirements, requirement analysis.

Introduction. Understanding the software requirements and knowledge of the software development process play a key role in studying and using a huge volume of information for software risk recognition. This information explains the risks nature. As common, risk is defined as the denial of one or more objectives or the loss of the achievement of some relevant objectives. Risks always blur targets, and certain goals can be risky. [1-5]

In developing an effective risk management model (RMM), it is important to consider risks from all technical and non-technical aspects of development. It is well-known that Software Development Live Cycle (SDLC) is almost standardised and has some predefined stages. We can face different namings in the literature for those stages but let's follow the namings from [1]. The suggested structure of 7 predefined SDLC stages [1] defines the order of software development processes and related activities including risk management.

Generally, the SDLC stages are the same for different Software Development Models (SDM) because they are development milestones and represent a logical sequence of the

executed tasks. As was shown in [1-4], there are a lot of different SDMs: Waterfall, Agile, Extreme Programming, Lean, Prototyping Methodology, Rational Unified Process, Dynamic Systems, Feature-Driven Development, Spiral Model, Joint Application, Scrum and Rapid Application. Each provided SDM is a good choice for some reasons and conditions. The authors in [1] did a good investigation and described these methodologies with explanations, advantages and disadvantages.

Indeed, each SDM predefines some risk management processes for different SDLC stages. The main goal of risk management is to identify and control all possible risks before they occur during software development. [5] Therefore, various types of risks can be detected at almost every stage. The comprehensive risk model helps manage risks throughout the project, increasing the likelihood of successful delivery. Developing a comprehensive risk model to manage software development project risks, we should revise and refine them iteratively if new risks appear. [5]

In [6] authors came out from a prioritised list of top ten software risk items and considered them through RMMl using unique features of a proposed audit component. The proposed software RMM [6] has 5 phases: risk identification, risk measurement, risk assessment, risk mitigation and contingency plan. This model enriched the main phases of the Boehm RMM together with a wider range of risk categories. They related the selected risk categories and the corresponding factors by preparing a classifier. As the authors proved in [6], the proposed model reduces the unforeseen risks or risks that have already occurred by creating a verifier core that comprises risk managers and experts. The verifier core is dynamic as it can adapt to each phase, and this makes the management process efficient and up-to-date.

The RMM is specifically tailored for software development projects [7] and has useful relationships with the “Functional requirement analysis” step and the “Changing project plan” and so far for “Establishing the scope of software development project”. The image illustrates a RMM. This model integrates various decision points and processes to ensure comprehensive management of risks throughout the SDLC. The provided solution in [7] has decision-making attributes but doesn’t cover the entire SDLC and of course, we have no connection with SDM in particular use cases.

Concurrently, throughout the project lifecycle, there is a continuous cycle of learning and adjusting. Functional requirements are analyzed and refined, and lessons learned are integrated into the project processes to enhance future decision-making and risk management. By the way, this model [7] highlights a systematic and dynamic approach to risk management, ensuring that risks are identified, analyzed, and mitigated effectively to support the successful completion of software development projects which are pretty good targets for any risk management system.

A proposed method in [8] of modelling the risks of software development makes it possible to assess different situations at the stages of SDLC, as well as to develop a strategy and tactics for predicting, perceiving and overcoming the negative consequences of their manifestation. The model determines the average value of the probability of potential risk events when developing a suitable set of software that is useful for formulating classification rules of potential risk events according to the probability of their occurrence. [8]

Technical issues directly related to system hardware and software, such as tool support, development platform, the technical complexity of the project, specific device or hardware, and performance characteristics of the product to be developed and deployed. Non-technical issues relate to the organizational environment, project implementation, development process, methodological and management issues. [9]

The authors in [9] view goals as goals, expectations, and limitations of technical and non-technical challenges and risk factors that prevent the achievement of those goals. Details of early development components and project success factors are primary sources that facilitate the understanding of construction fundamentals. These suggestions we considered as useful ones and will rely on them in the future.

In [10], the authors present RMM to effectively address the risks that obstruct successful project outcomes. The approach explicitly models the relations between the goals based on the software development components and project success indicators with the risk factors that obstruct these goals which is very interesting for us.

As common, we have five basic steps taken to manage risks in software development. [6, 10] These steps are referred to as the risk management process. It begins with identifying risks, goes on to analyze risks, then the risk is prioritized, a solution is implemented, and finally, the risk is monitored. Each SDM has predefined software risk management processes.

The authors in [11] follow the goal of finding out whether there is a specific SDM which will be able to manage the predefined scope of risks. The research method used to accomplish this task was a comparative study. A comparative study is a method used to compare two or more ideas that have significant differences. [11] Separately, some authors have designed an error taxonomy, which includes risk indication activities based on appeared errors. [12] Authors in [13] have shown a relationship between risk management principles with SDMs.

The usage of indicators is effective in decision-making for risk management tasks. In fact, despite the relevance of risk management in software projects, software development organisations are commonly overlooked. One of the reasons for this fact is that the concept of risk is abstract and subjective, and its management does not bring obvious immediate practical results. [14] Thus, in this context, [14] aims to define and propose indicators that are specific to the environments of software projects to support risk assessment activities - risk identification and risk analysis. On par with risk identification and risk analysis, we still recognize treatment, monitoring and mitigation activities as described before.

According to [15], the world experience of risk management proves that the principle of applying a process approach has become the main principle of risk management modernization. Following the process approach and approved global standards, to ensure the effectiveness of the risk management system at enterprises, RMM is being built, which should include amidst main components: risk identification, risk analysis, risk treatment, risk monitoring and risk mitigation due to risk assessment values.

The context plays a crucially important role in the risk detection and management process due to the opportunities of Large Language Models (LLM). The authors in [16] have shown the applicability of risk recommendations for new projects based on the similarity analysis of contextual stories. This study applies context history inference to project design and planning, focusing on risk recommendations. Thus, with recommendations tailored to the characteristics of each new project, the manager begins with a broader set of information for more assertive project planning. [16] Using the situational approach, as was shown in [17], we can describe a subject domain of risks in software development.

Emerging technologies and innovations including programming languages, frameworks, and tools affect different risks at different SDLC stages. Of course, they bring new values and benefits but there are risks due to unknown potentials and a lack of appropriate expertise. Poorly formalised or frequently changing requirements increase risks, leading to budget overruns,

delays, and time with resource constraints. Thus, from time to time, we should update the risk management activities due to new requirements, new technologies in the market and so on.

After analysing the existing SDLC, SDMs and RMMs we notice that it is difficult to recognise software development risks based on requirements for software in the initial stages due to a huge variety of risks, a lack of information about innovative technologies and unclear context for risk appearance. Finally, the existing RMMs don't allow help to recognise the risks from the project requirements description. Thus, we propose our RMM, taking into account the benefits and shortcomings of the analysed models, which covers the software requirements, SDLC stages, SDM processes, risk management steps and related context. The software development risk taxonomy is being built on the proposed RMM.

Risk assessment for SDLC in conditions of different SDMs. From a software risk perspective, at each SDLC stage, we consider the characteristics of software risk groups, their indicators and their recognisability. It is a reason why we want to inspect each stage more deeply due to risks appearance and so on. Therefore, we use the advantages and disadvantages of each SDM in our analysis because we would like to improve the software development risk taxonomy.

The risk identification method is important place in a comprehensive risk management approach to improve project success. It is based on the software development risk taxonomy, which organises risks into a hierarchy of three levels: section, subsection, and group. The method includes a taxonomy-based questionnaire (TBQ) which consists of questions for each taxonomic group of risks designed to identify potential risks and issues affecting the software product. Involving some software development experts, who work at big IT companies, we asked them about the methodologies and risk levels at each stage of SDLC. We used the method of average arithmetic ranks to agree with the opinions of experts but will pay more attention to this problem in the next investigations. The gathered results are shown in Table 1.

Table 1. The results of the expert survey

Methodology/ Process	Ideation	Analysis	Design	Development	Testing	Deployment	Maintenance
Dynamic Systems Development	Low	Low	Low	Low	Low	Low	Low
Scrum Development	Low	Low	Low	Medium	Low	Low	Low
Extreme Programming	Medium	Low	Low	Low	Low	Low	Low
Agile Development	Medium	Low	Low	Medium	Low	Low	Low
Spiral Development Model	Medium	Low	Low	Medium	Low	Low	Low
Joint Application Development	Medium	Low	Low	Medium	Low	Low	Low
Lean Development	Low	Low	Medium	Medium	Medium	Low	Low
Rational Unified Process	Low	Medium	Medium	Medium	Medium	Medium	Medium
Feature-Driven Development	Medium	Medium	Medium	Medium	Medium	Medium	Medium
Prototyping Methodology	High	Medium	Medium	High	Medium	Medium	High
Waterfall Development	Low	Medium	High	High	High	Medium	Medium
Rapid Application Development	High	Medium	Medium	High	High	Medium	High

The explanation of risk levels:

- Low Risk (green colour). The methodology is well-equipped and organised to handle this stage with minimal potential risks.
- Medium Risk (yellowish colour). In this case, we expect certain troubles associated with this stage of the methodology that require careful management.
- High Risk (reddish colour). This stage presents significant challenges and risks under this methodology and requires extensive management and clarification.

According to our measuring (tabl. 1), we have a rough understanding of SDM and related risks in each stage of SDLC. These results allow us to understand the challenges of the chosen SDM at each stage of SDLC. Building the taxonomy of software development risks we will pay attention to these results and identify risk groups for reddish and yellowish SDLC stages.

Thus, we will continue to define the risk attributes around them but, before we go ahead with nested items of the taxonomy, we would like to formalise and create a software development RMM where we can connect functional and non-functional software requirements with software development processes and appropriate risks.

Then we tried to extend the number of risk attributes and risks themselves in the dataset but we faced the multi-label problem when one attribute could be related to some risks with special weight. Thus, we want to avoid this problem by general taxonomy because when we have taxonomy, we can split the same meaning risks and their attributes between different SDM and SDLC stages.

Risk indicators and related activities. Analysing the SDM process for risk handling we came to the risk indication activities for SDM. These activities allow the recognition of semantic relationships for each SDM at each SDLC stage. Under a **risk indication activity,** we understand an action which helps identify, analyse, treat, monitor and mitigate the risk. Continuously analysing the risk indication activities for other SDMs we gathered the most essential of them in Table 2 which describes a general SDM to how each method applies to risk management at each stage of the SDLC. This representation involves the common risk management steps: identification, analysis, assessment/classification, treatment and monitoring/review at each stage of the SDLC.

Generally, analysing the risk indication activities of risk management systems, we have recognised some applicable risk management processes for all the SDMs. For example, they are for two of the most popular SDMs: Waterfall Development and Agile Development.

The essential risk management processes in Waterfall Development:

- Creating comprehensive requirement documentation to reduce misunderstandings.
- Thorough reviewing to obtain formal approval before proceeding with requirements.
- Phase-wise testing to catch defects before moving to the next phase.
- Formalisation of the processes for managing changes and minimising risks.
- Perform detailed risk assessments before the end of each phase to have time for changes in the next phase.
- Gather post-implementation reviews after project completion for future projects.
- Develop and maintain a contingency plan for critical path activities.

The essential risk management processes in Agile Development:

- Regular retrospectives reflect the risks of the current working methods.
- Facilitate daily meetings to address current risks and issues quickly.

- Thoroughly review sprint outcomes and adapt the backlog to mitigate risks.
- Automate testing and deployment to detect integration issues as early as possible.
- User story mapping to ensure understanding and alignment across the team.

Table 2. Risk indication activities

Methodology / Stages	Ideation	Requirement Analysis	Architecture Design	Development	Testing	Deployment	Maintenance
Waterfall Development	Early risk identification and assessment to avoid scope changes later	Detailed risk analysis and evaluation in documentation	Architectural risks are identified and strategies formed	Code risks are treated through thorough planning. Integration risks evaluated and plans for testing established	Risk treatment through systematic testing	Deployment risks monitored and reviewed	Ongoing risk monitoring and maintenance strategies
Agile Development	Continual risk identification through feedback loops	Emphasis on user stories to identify and analyse risks.	Regular refactoring to address design risks	Continuous integration helps treat coding risks early. Daily builds and integration to manage integration risks	Sprint-based testing to treat and review risks	Incremental deployment to mitigate risks	Regular updates and retrospectives for maintenance risks
Extreme Programming	Risks identified through fast iterations	Close customer collaboration for risk analysis	Simple design principles to reduce design risks	Pair programming and test-driven development to treat coding risks. Continuous integration and shared codebases reduce risks	Emphasis on customer tests to evaluate risks	Small releases to manage deployment risks	Iterative improvements and constant feedback loops
Lean Development	Risk identification aligned with value stream mapping	Lean analytics to evaluate requirement risks	Design risks are managed by removing waste and inefficiency	Emphasis on automation and standardization for coding. Just-in-time integration to minimize risks	Built-in testing to continuously evaluate risks	Lean approaches to streamline deployment	Kaizen (continuous improvement) for maintenance
Prototyping Methodology	Prototyping to quickly identify feasibility risks	Risk analysis through iterative feedback on prototypes	Design risks identified and treated with each prototype iteration	Coding in stages, with risks identified in prototype reviews. Integration handled iteratively to manage risks	Prototype testing to evaluate risks continuously	Early and frequent deployment of prototypes	Feedback incorporated into ongoing maintenance
Rational Unified Process	Risks identified during the inception phase	Requirements elaborated with risk considerations	Architectural baseline to address design risks	The implementation phase focuses on managing coding risks. Integration assessed at each iteration	Testing phases specifically aimed at risk treatment	The transition phase handles deployment risks	Support and maintenance are planned and risk-aware
Dynamic Systems Development	Feasibility studies to identify initial risks	Iterative workshops to analyze requirement risks	Design risks managed through continuous user feedback	Repeated prototyping helps treat coding risks. Frequent integration sessions to manage integration risks	Demonstrators and prototypes used to test and adjust	Deployment reviewed through user feedback	Continuous user involvement helps monitor maintenance risks
Feature-Driven Development	Initial risk assessment during overall model creation	Feature list helps analyze requirement risks	Design by feature to address design-specific risks	Coding by feature, focusing on risk mitigation per feature. Regular builds to manage integration risks	Feature-based testing to evaluate risks	Features deployed incrementally to manage risks	Ongoing feature enhancement to address maintenance risks
Spiral Development Model	Objective setting includes risk determination	Progressive risk analysis and requirement refinement	Prototyping and simulations to address design risks	Development and testing to manage coding risks. Risk-driven integration process	Detailed risk evaluations during test phases	Systematic risk management for deployment	Regular risk reassessment for maintenance
Joint Application Development	Risks identified in collaborative sessions	Requirement workshops to analyze and evaluate risks	Design sessions help identify design risks early	Coding in collaborative environments to treat risks. Integration tested in joint sessions	Testing involves all stakeholders to review risks	Deployment planned with stakeholder input to minimize risks	Continuous feedback loops for maintenance risks
Scrum Development	Sprint planning includes risk identification	Backlog grooming sessions to analyze risks	Design risks managed during sprints	Coding in sprints with continuous reviews for risks. Integration risks handled during sprint reviews	Sprint-based testing phases for risk evaluation	Deployment risks are managed at the end of sprints.	Sprint retrospectives to monitor maintenance risks
Rapid Application Development	The initial phase includes risk identification for the scope	Workshops and prototyping to analyze requirements risks	Iterative design and feedback loops to manage design risks	Timeboxing coding phases to quickly address risks. Integration is conducted in stages to evaluate risks	Testing phases focus on treating identified risks	Staged deployment to mitigate deployment risks	Ongoing iterations for maintenance with risk reviews

Stakeholders, architects, experts and others are proactive participants in risk management discussions. They define the problem statement for the business problem at the first SDLC stage named "Project initiation and planning". At this stage, we create an initial solution which is a matter for discussion at the "Architectural Design" and "Requirement analysis" SDLC stages. After, we try to identify the risks as early as possible even when we have an initial solution. The relationships between the 7 stages of SDLC and the 5 steps of risk management processes are shown in Figure 1.

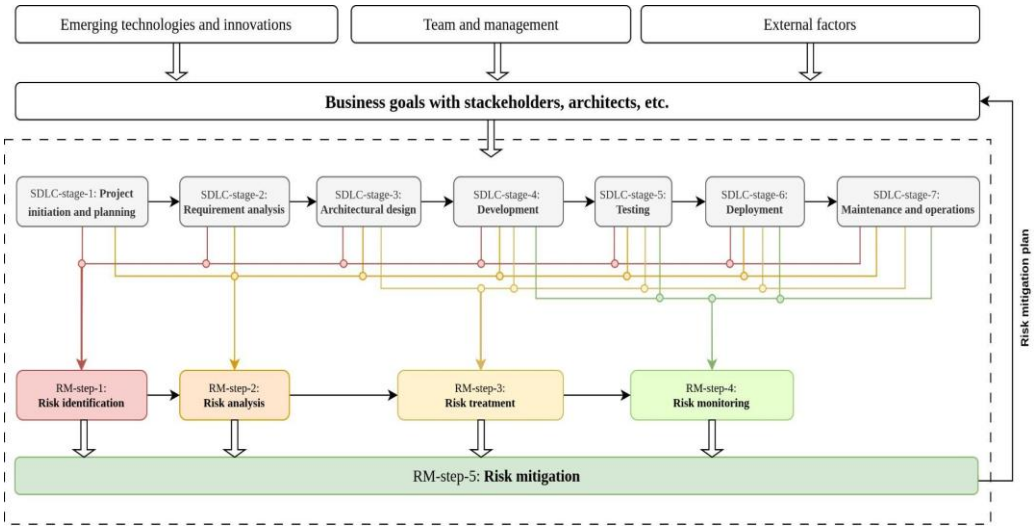

Fig. 1. Relationships between SDLC stages and RM steps

During all 7 stages of SDLC, we continuously pay attention to the new possible risks. In addition, we check the status of already identified risks in the stages from "Development" to "Maintenance and operations" but for two reasons: treatment and monitoring. The difference between treatment and monitoring is a chosen algorithm for risk assessment. "Risk control" is the final risk management stage of creating a risk mitigation plan for stakeholders and other participants for analysis. Understanding these relationships allows us to drill down deeper into the nested classes in the taxonomy correctly due to behaviour scenarios. Now, if we want to add a new scenario we will have the right place for taxonomy expansion.

Emerging technologies, innovations and other external factors impact the software development processes inside projects at each stage of SDLC. Team and management make different impacts according to chosen SDMs but are slightly stable at each SDLC stage.

Software development RMM. Considering the software requirements, SDLC stages, SDM processes and RM steps the proposed model emphasises the interdependence of the various elements of the risk management process because, it is important to have an all-in-one approach that covers all aspects of project development, from planning to deployment and maintenance. The RMM describes a structured situational approach to risk management in the software development processes, bringing together various elements, from stakeholders to technology and external factors. The benefits of the situational approach are obvious because we created a software development RMM (fig. 2) for risk processing.

Stakeholders and risk managers are the main drivers of the project's goals and specifications with the risk management process, and who is responsible for formalising of the project scope with a list of requirements.

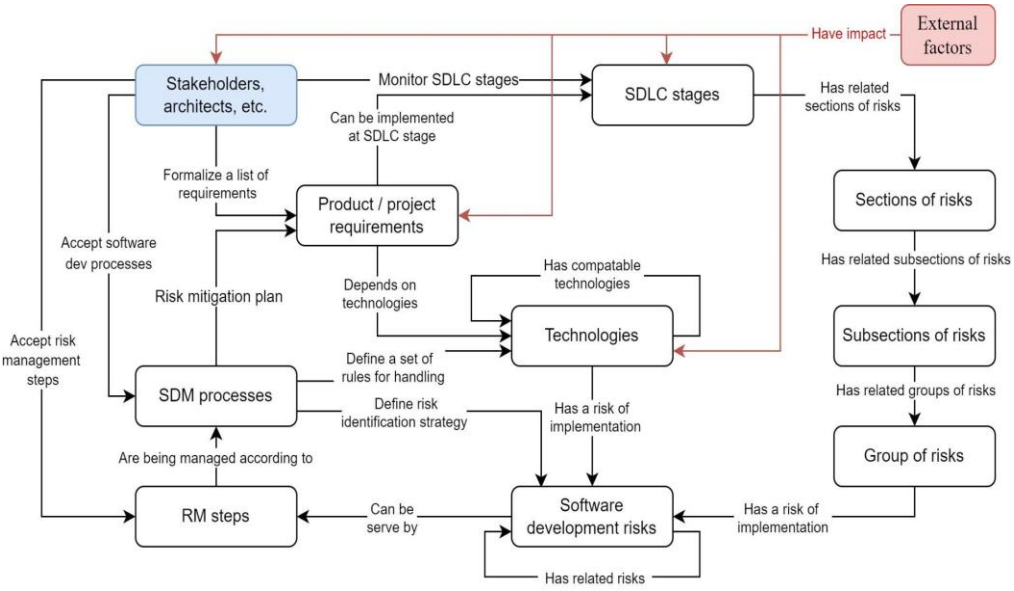

Fig. 2. Software development RMM

Today, the software development process is complex because of different technologies in the background. The mix of technologies brings us benefits but risks as well. The project's risks, development processes and external factors require proper management of these risks.

The development processes are being controlled according to chosen SDM processes. The SDM processes are being approved by stakeholders. They define a risk mitigation plan and curate the chosen technology. The efficiency of risk management protocols depends on the efficiency of the chosen SDM for the project implementation and chosen risk management steps. Proper execution of these steps ensures that risks are managed systematically and effectively.

During the SDLC stages, we consider three levels of risks. On the top level, we have sections. Risk in the section has more global effects on the project and can have subrisks. Meanwhile, in the section, we can have different subsections which can include still group levels. The number of levels doesn't affect strongly and depends on the actual risk's components. Sometimes we can split risk into small components for easy calculation but sometimes we should describe a complex risk. We accept 3 levels based on the results of investigations.

As can be seen from Figure 2, the external factors influence the project team, such as market trends, regulatory changes, or economic conditions. They have a significant impact on the project and must be factored into the risk management plan.

Software development risk taxonomy. The context is correctly represented by a situational approach where all risk indication activities and risk mitigation algorithms are in the same context with software requirements at the SDLC stages for different SDMs. The software requirements are the first item there. Indeed, the software development risk taxonomy,

classifying the risks, helps us process the risks much better, builds the risk datasets and knowledge base and so on. It works fine when we can correctly identify a risk. Sometimes, one risk might have some indicators or risk indicators can point to some risks. In this case, we should dive deeper to find the semantic relationships between indicators and requirements and include them in the taxonomy (fig. 3).

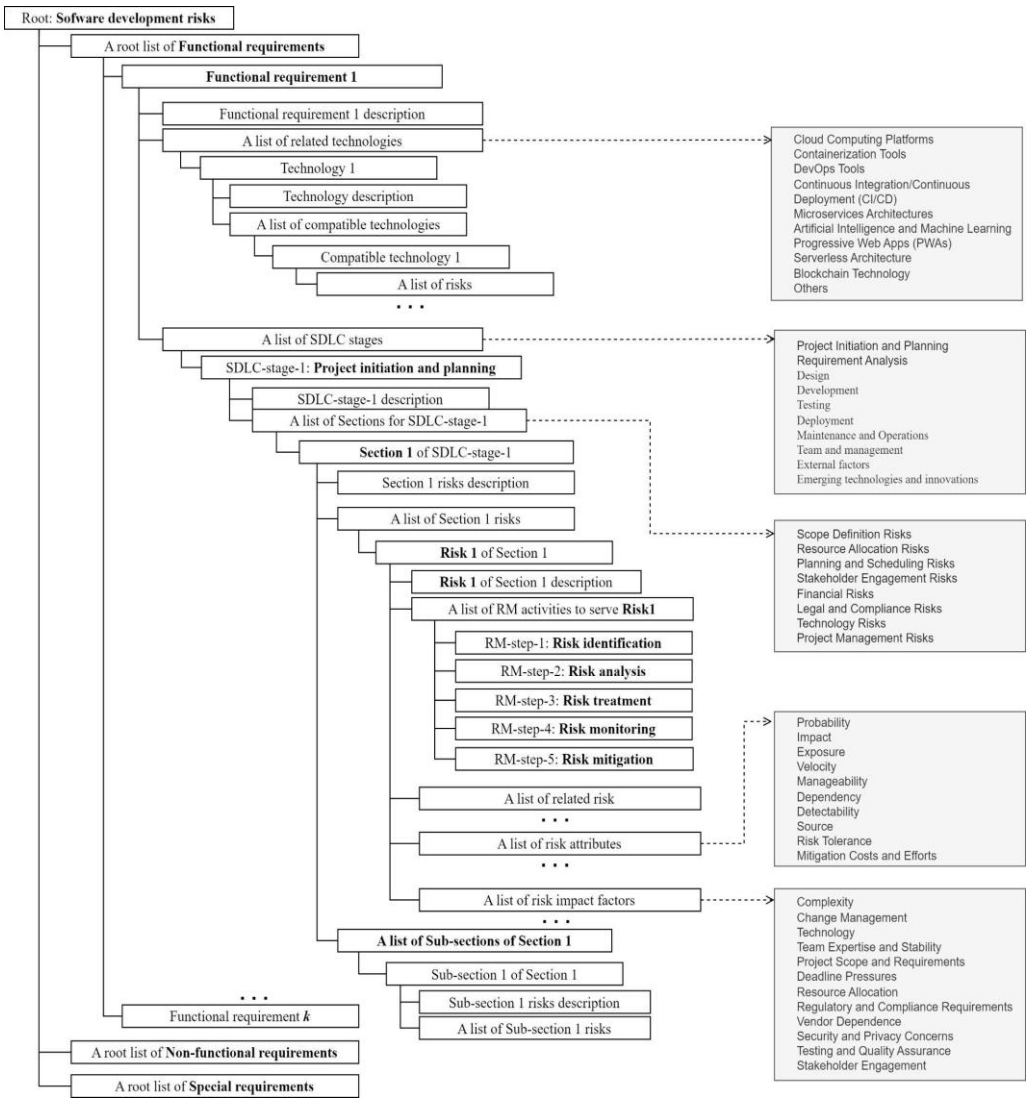

Fig. 3. A high-level fragment of the Software Development Risk Taxonomy

Software development risks are affected by various factors, including complexity, change management, technology, team experience, project scope and requirements, time constraints,

resource allocation, regulatory and compliance requirements, vendor dependency, security and privacy issues, testing processes, and quality assurance and stakeholder involvement.

Complex systems often have more integration points and dependencies, which can create more potential points of failure. Change management is critical because frequent changes without proper control can lead to inconsistencies, defects, and project delays.

Technology, including programming languages, frameworks and tools, impacts risk. New or rapidly evolving technologies can offer benefits but also carry risks due to potential unknowns and a lack of established best practices. The expertise and stability of the team are essential to the success of the project. Poorly defined or frequently changing requirements can increase risks, leading to budget overruns, delays and resource constraints. Deadline pressures can force teams to rush through development stages, increasing the risk of errors and failures. Insufficient or poorly allocated resources can hinder project completion and increase risks, related to quality and deadlines.

Summary. A software development risk taxonomy is one of the important parts of intellectual systems for risk management. The developed taxonomy combines software requirements, SDLC stages and risk management steps for different SDMs. It facilitates a structured and systematic approach to risk management improving the decision-making processes. Being developed based on domain experts' knowledge, the taxonomy allows us to follow the best practices, using existing expertise, in the poorly formalised field of software development.

Among the received results we want to highlight:

- Now, SLDC stages, SDM processes, RM steps and software requirements are connected within the newly updated software development RMM.
- The software development risk taxonomy conception is being enhanced in place of software requirements, SLDC stages, SDM processes and RM steps.
- The first version of the created taxonomy includes 793 different requirements, 10 sections of risks, 64 subsections and 415 groups of software development risks according to the proposed taxonomy structure. The experts continue to work on the taxonomy increasing the number of requirements and risks for them.

The created RMM and appropriate taxonomy allow systematic identification of the risks across different sections, subsections and groups. This structure helps ensure that risks are not overlooked and that every potential threat is considered during the risk assessment phase. The number of levels in the risk hierarchy could be extended in the future if it is required.

By defining the hierarchy of risks we enhanced risk understanding and clarity. It helps to increase the decision-making process for a clearer understanding of potential issues. This clarity is crucial for effective communication and ensuring that everyone has a clear understanding of risk. It is crucially important for risk mitigation purposes because the taxonomy helps to define the nature of the identified risks precisely.

The taxonomy splits different risks by meaning and allows leveraging LLMs to automate many aspects of risk management, from identification to monitoring. Of course, if new risks appear, a flexible taxonomy can be updated and expanded allowing LLMs to adapt quickly to the changes in the project environment or technology avoiding unpredictable hallucinations. This adaptability allows immediate reaction to new risks that appear as technology evolves.

In conclusion, the proposed RMM and appropriate risk taxonomy significantly enhance the capabilities of LLMs in managing software development risks by providing the context for risk

appearance and a structured and consistent framework for risk management. This allows us to manage future risks starting from the requirement formalisation and description stage.

References

[1] *Hossain, Mohammad*. (2023). Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management. DOI: https://doi.org/10.36948/ijfmr.2023.v05i05.6223

[2] *Hrishitva Patel*. An Insight on (SDLC) Software Development Lifecycle Process Models. Advance. April 21, 2023. DOI: https://doi.org/10.31124/advance.22354453.v1

[3] *Rozhnova, T., Tomachynska, V., & Korsun, D*. (2022). Life cycle models, principles and methodologies of software development. Scientific Collection «InterConf+», I. 28 (137), pp. 394–401. DOI: https://doi.org/10.51582/interconf.19-20.12.2022.040

[4] *Gurung, Gagan & Shah, Rahul & Jaiswal, Dhiraj*. (2020). Software Development Life Cycle Models - A Comparative Study. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 30-37. DOI: http://dx.doi.org/10.32628/CSEIT206410

[5] *M. Lyashkevych, V. Lyashkevych and R. Shuvar*. "Risks' Attribute Values Evaluation in Software Engineering by Monte Carlo Simulation," 2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT), Lviv, Ukraine, 2023, pp. 137-141. DOI: http://dx.doi.org/10.1109/ELIT61488.2023.10310775

[6] *Khatavakhotan, Ahdieh & Ow, Siew*. (2015). Development of a software risk management model using unique features of a proposed audit component. Malaysian Journal of Computer Science. 28. 110-131. URL: https://www.researchgate.net/publication/281993369_Development_of_a_software_risk_management_model_using_unique_features_of_a_proposed_audit_component

[7] *Dey, Prasanta & Kinch, Jason & Ogunlana, Stephen*. (2007). Managing risk in software development projects: A case study. Industrial Management and Data Systems. 107. 284-303. DOI: http://dx.doi.org/10.1108/02635570710723859

[8] *Y. Hrytsiuk, P. Grytsyuk, T. Dyak and H. Hrynyk*. "Software Development Risk Modeling," 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2019, pp. 134-137, doi: http://dx.doi.org/10.1109/STC-CSIT.2019.8929778

[9] *S. Islam and S. H. Houmb*. Integrating risk management activities into requirements engineering. In Proc. of the 4th IEEE Research International Conference on Research Challenges in Information Science(RCIS2010), Nice, France, 2010

[10] *S. Islam and S. H. Houmb*. Towards a framework for offshore outsource software development risk management model. Journal of Software (JSW), Special Issue: Selected Papers of the IEEE International Conference on Computer and Information Technology (ICCIT 2009), 2011.

[11] *Henri, Evans*. (2020). A Review of Risk Management in Different Software Development Methodologies.

[12] *Agrawal, T., Walia, G.S. & Anu, V.K*. Development of a Software Design Error Taxonomy: A Systematic Literature Review. SN COMPUT. SCI. 5, 467 (2024). DOI: https://doi.org/10.1007/s42979-024-02797-2

[13] *Oehmen, Josef & Seering, Warren & Bassler, Denis & Ben-Daya, Mohamed*. (2013). A comparison of the integration of Risk management Principles in Product Development Approaches. 3.

[14] *Menezes Júnior, Júlio & Gusmao, Cristine & Moura, Hermano*. (2013). Defining Indicators for Risk Assessment in Software Development Projects. CLEI Electronic Journal. 16. 11-11.
URL: https://www.researchgate.net/publication/317447281_Defining_Indicators_for_Risk_Assessment_in_Software_Development_Projects

[15] Matusova, Olena & Victoriya, Andryeyeva & Viktor, Ahodzinsky. (2019). Risk Management Models. Herald of Kyiv National University of Trade and Economics. 128. 75-85.
URL: https://www.researchgate.net/publication/338171666_RISK_MANAGEMENT_MODELS

[16] *Alexsandro Souza Filippetto, Robson Lima, Jorge Luis Victória Barbosa*. A risk prediction model for software project management based on similarity analysis of context histories. Information and Software Technology. Volume 131. 2021. DOI: http://dx.doi.org/10.1016/j.infsof.2020.106497

[17] *Lyashkevych V.Y.* Using the situational approach in the construction industry ontology "Predictive diagnostics computer means" [Text] // V.Y. Lyashkevych, R.I. Makarchuk, A.A. Nadyeyev / Bulletin Khmelnytsky National University. - No 5. - 2013. - P. 152-158.

СТВОРЕННЯ ТАКСОНОМІЇ РИЗИКІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ КОМПЛЕКСНИХ ПРОЦЕСІВ РОЗРОБКИ

М. Ляшкевич, І. Рогацький, В. Ляшкевич, Р. Шувар

Кафедра системного проектування,
Львівський національний університет імені Івана Франка,
вул. Драгоманова, 50, Львів, 79005, Україна
mariia.liashkevych@lnu.edu.ua

Ризики програмного забезпечення завжди є надзвичайно важливою темою для досліджень, оскільки процес розробки програмного забезпечення досить дорогий, а конкуренція досить висока, щоб його ігнорувати. Хоча «золота» ера для стартап-проектів повільно закінчується, останні досягнення в генеративному штучному інтелекті показують, що саме час «ризикнути» і захопити ринок програмного забезпечення за допомогою цієї технології. Таким чином, необхідно проаналізувати вже відомі ризики та визначити нові ризики, пов’язані з бізнес-моделями та ринковими умовами з генеруючою здатністю ШІ.

Взаємозв’язок між вимогами програмного забезпечення та ризиками розроблення програмного забезпечення встановлені за допомогою визначених індикаторів ризиків. Запропоновані індикатори ризиків на кожній стадії життєвого циклу розроблення програмного забезпечення та їх зв’язки із обраною методологією розроблення програмного забезпечення дають можливість краще структурувати поняття в таксономії ризиків розроблення програмного забезпечення.

У статті проаналізовано уже існуючи таксономії програмних ризиків, їх недоліки та переваги, розглянуто стадії життєвого циклу розробки програмного забезпечення, активності щодо керування ризиками в умовах різних моделей розроблення програмного забезпечення. Використовуючи запропоновану таксономію, пов'язано всі вище згадані

активності та процеси в одній таксономі, що дозволяє легко ідентифікувати ризики на основі відомих вимог до програмного забезпечення та навпаки.

Створена таксономія була підтверджена деякими експертами предметної області. ChatGPT4 є одним із експертів, які розраховують на можливості великих мовних моделей для вирішення завдань узагальнення та класифікації тексту. Для автоматичного опрацювання запропонованої таксономії засобами великих мовних моделей всі поняття таксономії ризиків маю відповідний текстовий опис у загальному словнику понять. Практичні результати таксономії ризиків є надзвичайно важливими, оскільки ми уникаємо галюцинацій великих мовних моделей і застосовуємо керований підхід на основі таксономії до швидкої розробки для управління ризиками.

Ключові слова: ризики розробки програмного забезпечення, таксономія ризиків, розпізнавання ризиків, виявлення ризиків, таксономія, вимоги до програмного забезпечення, аналіз вимог.

The article was received by the editorial office on 22.07.2024.
Accepted for publication on 01.08.2024.