Method of the Software Risks Management

Tetiana Hovorushchenko[1]

[1] Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine

Abstract

The result of any software project depends on the number and magnitude of risks of insufficient software functionality, non-compliance with project deadlines, budget overruns. Therefore, risks management should be one of the foundations of project management, and the actual task now is improving the risks management in software development. The main task of this study is detailing and formalizing the method of risks management in software development. The paper proposes a method of the software risks management, which allows identifying sources of risks and possible risks for any software project, as well as to assess risks, determining their priority and measures to reduce or eliminate risks. In addition, the method allows risks assessment after the application of selected measures to reduce or eliminate risks, which makes it possible to select the best measure to minimize the magnitude of each risk. The presented method provides a mathematical basis for a risks management process, which reduces the complexity and increases the effectiveness of risk management. The prospect for further research by the authors is to develop a risk management system in the software development, which will be based on the proposed in this paper method.

Keywords

Risks in software development, risks management in software development, method of the software risks management, risks identification, risks analysis, risks planning, risks monitoring.

1. Introduction & Related Works

At present, despite the rapid development of the software engineering industry, a significant number of software projects remain that cannot be considered completely successful. The success of the software project means the timely implementation of the program project within the allocated budget and with the implementation of all necessary capabilities and functions [1-5].

Statistics on the success of software projects for 1994-2019, presented by The Standish Group International (CHAOS report) [1-5], gave the opportunity to see an increase in the number of successful projects and a decline in the number of failed projects in 2010-2019, while the share of problem projects is fairly stable in 2006-2019 and accounts for about 50% of projects.

CITRisk'2021: 2nd International Workshop on Computational & Information Technologies for Risk-Informed Systems, September 16–17, 2021, Kherson, Ukraine
EMAIL: tat_yana@ukr.net (T. Hovorushchenko)
ORCID: 0000-0002-7942-1857 (T. Hovorushchenko)

Statistics [1-5] also show that only 16% of software projects are successfully completed by medium-sized companies on time and budget. The situation with large companies is much worse – only 9% of projects are invested on time and budget. Projects implemented by the largest American companies have about 42% of the functionality offered in the initial stages. Smaller companies do better: 78.4% of projects implement 74.2% of their planned functionality.

Research by McKinsey & Company [6] in collaboration with the University of Oxford also found that half of large-scale software projects with a total budget of more than $ 15 million significantly exceeded planned costs, including: average project overruns are 66%, average project time overruns are 33%, and the average number of profit losses is 17%.

Thus, software development is not always successful and is often associated with the risks of insufficient software functionality, non-compliance with project deadlines or budget overruns [7-11]. Risks are negative events of a probabilistic nature that negatively affect the outcome of the project; negative events and their magnitudes that reflect losses and damages from processes or products caused by defects in the design of requirements, by shortcomings in the justification of software projects, as well as in the subsequent stages of development, implementation and all software lifecycle [12-14]. Risks are manifested as possible negative consequences or losses during the operation of the software, as negative consequences of the operation or violation of the security of the software as a result of deviation of the characteristics of objects or processes from the specified customer requirements, which can cause the damage to the system, the external environment or the user (for example, loss of the system, loss of consequences of the person or team activity, personal damage or the emergence of legal liability for negative project results) [15, 16].

Risk is a probable event that may or may not occur. The causes of occurrence and manifestation of risks can be: malicious, active influences of stakeholders or accidental negative manifestations of defects of the environment, system, actions of developers or users [17-19].

The risks of the accidental negative effects of defects in the absence of malicious effects on the system depend on failure situations that affect the workability and security of their basic functions realization, which can be caused by defects and anomalies in hardware, software, data or computational processes [17-19]. This significantly distorts the process of functioning of the systems, which can cause significant damage when using systems. The main sources of failure situations are incorrect initial design requirements, hardware failures and faults, defects or errors in software and data. Currently, there are no methods, which provide to guarantee the absence of defects either in the specifications, or directly in the programs, or in the operating documentation. From the end user's point of view, the manifestations of software defects can range from temporary inconveniences to man-made disasters. In real complex systems, catastrophic consequences and failures with large losses are possible, which may exceed the consequences of malicious influences, so such risks require adequate methods and means to minimize them [20, 21].

In general, the following typical important reasons can be identified, which lead to the emergence of risk situations of the second type in the software projects: unrealistic assessment of the required time of project implementation and the allocated budget; unrealistic assessment of the capabilities of the development team; insufficient number and qualification of the development team; insufficient ability to use the tools by developers; errors in determining the requirements for the developed software (including insufficient detailing of requirements); violation of the basic rules of development processes (for example, violations in version control, which lead to the loss of versions); continuous change of requirements to the developed software during the project; a significant change in the market situation, which makes it meaningless to follow the original plans (for example, the emergence of affordable software on the market, which exceeds the capabilities of the developed software); continuous change of "rules of the game" in the development team or project group (rules of communication, division of responsibilities, segregation of duties); software architecture design errors; software development errors; integration errors; shortcomings of external service; technical and software failures [22-24].

There are three classes of risks in the software lifecycle:

- deficiencies and defects of functional suitability – distortion or incomplete implementation of the desired purpose, functions or interaction of software with the components of the system or the environment
- insufficient and non-compliant with the requirements the implementation of the design characteristics of the quality of the software during its operation and use for its intended purpose
- violation of restrictions on the use of economic, time or technical resources in the creation and use of software [25].

The task of developers is to reduce and eliminate risks. Reducing the risks of a software project helps to increase its success, quality, efficiency and effectiveness. Therefore, *the actual task* now is improving the risks management in software development.

For the successful implementation of software projects, one of the foundations of project management is risks management, which covers the entire software life cycle. Risks management is the process of making and implementing management decisions aimed at reducing the likelihood of an adverse outcome and minimizing possible losses caused by its implementation; these are systematic processes related to the identification, analysis and decision-making, which ensure the minimization of the negative consequences of the occurrence of risks events, as well as maximizing the probability and consequences of the occurrence of positive events [26, 27]. Risks management includes a full understanding of the internal and external causes that affect the project and may lead to its failure. Risks analysis is performed after the formation of the project plan. The main purpose of risks management is the identification and control of factors that are rare and lead to project variations.

There are various models of risks management [26, 27], the most used of which is the model of the Software Engineering Institute (SEI), which includes both the requirements of standards and known "best practices" of risks management. The SEI model is presented in the form of textual recommendations and a plan; there is no formalized method of risk management, which leads to the free use and interpretation of this model.

From the results of the analysis of the current state of the software development industry it follows that a promising area of research is the development of a mathematical method of risks management in software development. Therefore, *the main task of this study* is detailing and formalizing the method of the software risks management.

2. Method of the Software Risks Management

The method of the software risks management consists of the following stages:

Stage 1. Risks identification:

- Identification of possible sources of risks – let's present the 18 most common sources of risks in the form of the following set: PSR = {psr_1, ..., psr_{18}}, where psr_i – possible source of risk (i = 1..18), namely: psr_1 – functional characteristics, psr_2 – quality characteristics, psr_3 – reliability characteristics, psr_4 – applicability, psr_5 – time performance, psr_6 – maintainability, psr_7 – reuse of components; psr_8 – limitation of the total budget, psr_9 – unavailable project cost, psr_{10} – low degree of realism in estimating project costs; psr_{11} – properties and possibilities of flexibility of change of plans, psr_{12} – possibilities of violation of the established terms of stages of a life cycle, psr_{13} – low degree of realism of plans and stages of a life cycle; psr_{14} – project strategy, psr_{15} – project planning, psr_{16} – project evaluation, psr_{17} – project documentation, psr_{18} – project forecasting; herewith psr_1-psr_7 belong to the sources of technical risks, psr_8-psr_{10} belong to the sources of cost risks, psr_{11}-psr_{13} belong to the sources of plan risks, psr_{14}-psr_{18} belong to the sources of risks of project management processes and procedures.

The rules for determining the sources of risk are as follows:

if the software documentation has no functional characteristics or there are unrealistic or invaluable functional characteristics, then $psr_1 =1$, else $psr_1 = 0$;

if the documentation does not contain quality characteristics or there are unrealistic or invaluable quality characteristics, then $psr_2 =1$, else $psr_2 = 0$;

if there are no reliability characteristics in the documentation or there are unrealistic or invaluable reliability characteristics, then $psr_3 =1$, else $psr_3 = 0$;

if the documentation does not contain recommendations for the future applicability of the software, then $psr_4 =1$, else $psr_4 = 0$;

if the documentation lacks the characteristics of time performance or there are unrealistic or invaluable characteristics of time performance, then $psr_5 =1$, else $psr_5 = 0$;

if the documentation does not contain recommendations for future software maintenance, then $psr_6 =1$, else $psr_6 = 0$;

if there are no component reuse proposals in the documentation or there are unrealistic or invaluable component reuse proposals, then $psr_7 =1$, else $psr_7 = 0$;

if there are restrictions on the total budget in the specification, then $psr_8 =1$, else $psr_8 = 0$;

if the documentation indicates the unavailable cost of the project, then $psr_9 =1$, else $psr_9 = 0$;

if the documentation has a low degree of realism in estimating the cost of the project, then $psr_{10} =1$, else $psr_{10} = 0$;

if the documentation does not contain the properties and possibilities of flexibility to change plans or there are unrealistic or invaluable properties and possibilities of flexibility to change plans, then $psr_{11} =1$, else $psr_{11} = 0$;

if in the documentation there are possibilities of violation of the established terms of stages of a life cycle, then $psr_{12} =1$, else $psr_{12} = 0$;

if the documentation has a low degree of realism of plans and stages of the life cycle, then $psr_{13} =1$, else $psr_{13} = 0$;

if there is no project strategy in the documentation or there is an unrealistic or invaluable project strategy, then $psr_{14} =1$, else $psr_{14} = 0$;

if there is no project planning or there is unrealistic or invaluable project planning, then psr_{15} =1, else psr_{15} = 0;

if there is no project evaluation or there is an unrealistic project evaluation, then psr_{16} =1, else psr_{16} = 0;

if there is no project documentation, then psr_{17} =1, else psr_{17} = 0;

if there is no forecast of project success or there is unrealistic or invaluable project forecasting, then psr_{18} =1, else psr_{18} = 0;

if$(psr_1 = 1) \cup (psr_2 = 1) \cup (psr_3 = 1) \cup (psr_4 = 1) \cup (psr_5 = 1) \cup (psr_6 = 1) \cup (psr_7 = 1)$, then there are technical risks;

if $(psr_8 = 1) \cup (psr_9 = 1) \cup (psr_{10} = 1)$, then there are cost risks;

if $(psr_{11} = 1) \cup (psr_{12} = 1) \cup (psr_{13} = 1)$, then there are plan risks;

if $(psr_{14} = 1) \cup (psr_{15} = 1) \cup (psr_{16} = 1) \cup (psr_{71} = 1) \cup (psr_{18} = 1)$, then there are risks of project management processes and procedures.

- Identification of potential risks events - identification of all factors of anxiety and concern associated with the project, as well as constant consideration of other possible concerns, as the real problem at this stage is the risks that could not be identified. Based on the leading industry publications [10-25] let's form a set of potential risks events: PRE = {pre_1, ..., pre_{43}}, where pre_j – potential risk event (j = 1..43), namely: pre_1 – delays in supply of equipment required for the software development process, pre_2 – delays in the supply of software tools required to support the software development process, pre_3 – reluctance of developers to use lifecycle support software tools, pre_4 – rejection of CASE-tools, pre_5 – requests for more powerful tools of software development, pre_6 – insufficient performance of database(s), pre_7 – reusable software components have defects and limited functionality, pre_8 – inefficiency of software code generated by CASE tools, pre_9 – inability to integrate CASE tools with other tools project support, pre_{10} – the rate of detection of defects in the system below the previously planned rate, pre_{11} – defective system components; pre_{12} – underestimation of project costs (excessively low cost), pre_{13} – overestimation of project costs (excessively high cost), pre_{14} – financial difficulties for the developer's company, pre_{15} – reduction of the project budget during its implementation, pre_{16} – high cost of reworks required due to changing requirements, pre_{17} – reorganization of the development company; pre_{18} – changes in the work schedule, pre_{19} – violation of the work schedule, pre_{20} – the need to change many requirements, pre_{21} – the need for a large number of repeated works, pre_{22} – underestimation of project time, pre_{23} – overestimation of project time, pre_{24} – software size exceeds the planned size, pre_{25} – the size of the software is much smaller than the planned size, pre_{26} – the appearance on the market of similar software before the release of the developed software, pre_{27} – the appearance on the market of more competitive software; pre_{28} – low morale of staff, pre_{29} – weak interaction between members of the development team, pre_{30} – passivity of the project manager, pre_{31} – insufficient competence of the project manager, pre_{32} – customer dissatisfaction, pre_{33} – insufficient number of professionals with the required professional level, pre_{34} – illness of a leading developer at the most critical time, pre_{35} – simultaneous illness of several developers, pre_{36} – inability to organize the necessary staff training, pre_{37} – change of priorities in project management, pre_{38} – underestimation of the required number of developers, pre_{39} – overestimation of the required number of developers, pre_{40} – excessive project documentation, pre_{41} – insufficient project documentation, pre_{42} – unrealistic forecasting of project results, pre_{43} – insufficient professional level of developers; herewith pre_1-pre_{11} belong to potential technical risk events, pre_{12}-pre_{17} belong to potential cost risk events, pre_{18}-pre_{27} belong to potential plan risk events, pre_{28}-pre_{43} belong to potential risk events of project management processes and procedures.

The rules for determining the risks for a particular software project are as follows:

if delays in supply of equipment required for the software development process are possible, then pre_1 = «delays in supply of equipment required for the software development process», else $pre_1 = 0$;

if delays in the supply of software tools required to support the software development process are possible, then pre_2 = «delays in the supply of software tools required to support the software development process», else $pre_2 = 0$;

…

if project team includes developers with insufficient professional level, then pre_{43} = «insufficient professional level of developers», else $pre_{43} = 0$.

The rules for forming the set RSP $=\{rsp_1,\ldots,rsp_k\}$ of risks of a particular software project are as follows:

if $pre_1 \neq 0$, then: $k=1$, $rsp_k = pre_1$, $k = k+1$;

if $pre_2 \neq 0$, then: $rsp_k = pre_2$, $k = k+1$;

…

if $pre_{43} \neq 0$,then $rsp_k = pre_{43}$.

Stage 2. Risks analysis:

- Determining the probability of risk (probability of occurrence of a risk event). For each risk from a set RSP, the development team must determine the probability of its occurrence in the range [0;1]. The set of probabilities of risks has the form: $PR=\{pr_1,\ldots,pr_k\}$, where k is the number of risks of a particular software project.

The rules for classifying risks according to their probabilities are as follows (threshold values for establishing the risk category are formed as a result of analysis of industry publications [10-25]):

if $pr_h < 0.1$, then the probability of risk rsp_h is very low ($h = 1..k$);

if $(pr_h \geq 0.1)\cap(pr_h < 0.25)$, then the probability of risk rsp_h is low ($h = 1..k$);

if $(pr_h \geq 0.25)\cap(pr_h < 0.5)$, then the probability of risk rsp_h is medium ($h = 1..k$);

if $(pr_h \geq 0.5)\cap(pr_h < 0.75)$, then the probability of risk rsp_h is high ($h = 1..k$);

if $(pr_h \geq 0.75)$, then the probability of risk rsp_h is very high ($h = 1..k$).

- Determining the possible risk losses (how many losses. For each risk from the set RSP, the team of developers must set the amount of possible losses from its occurrence – in the range [0;1]. The set of risk losses has the form: LR=$\{lr_1,\dots,lr_k\}$, where k is the number of risks of a particular software project.

- Determining the magnitude of risk (mathematical expectation of damage). For each risk from the set RSP its magnitude of risk should be determined. The set of risk magnitudes has the form: MR=$\{mr_1,\dots,mr_k\}$, where k is the number of risks of a particular software project, $mr_i = pr_i \cdot lr_i$.
- Setting the priority level and ranking risks by priority. For establishing the level of priority and ranking of risks, let's find the maximal (mr_max) and minimal (mr_min) elements of the set MR. Let's further divide the received interval [mr_min; mr_max] at three intervals: $[mr\_min; mr\_min + \frac{mr\_max - mr\_min}{3})$, $[mr\_min + \frac{mr\_max - mr\_min}{3}; mr\_min +$ $+2 \cdot \frac{mr\_max - mr\_min}{3})$, $[mr\_min + 2 \cdot \frac{mr\_max - mr\_min}{3}; max]$.

The rules for identifying the level of priority of risks are as follows:

if ($mr_h \geq$ mr_min)$\cap$($mr_h < (mr\_min + \frac{mr\_max - mr\_min}{3})$), then the level of risk priority rsp_h is low (h = 1..k);

if ($mr_h \geq (mr\_min + \frac{mr\_max - mr\_min}{3})$)$\cap$($mr_h < (mr\_min + +2 \cdot \frac{mr\_max - mr\_min}{3})$), then the level of risk priority rsp_h is medium (h = 1..k);

if ($mr_h \geq (mr\_min + +2 \cdot \frac{mr\_max - mr\_min}{3})$)$\cap$($mr_h \leq$ mr_max), then the level of risk priority rsp_h is high (h = 1..k).

As a result of applying the above rules for identifying the level of priority of risks to all risks of the project we will have a set of priority risks (high priority), a set of secondary risks (medium priority) and a set of least risks (low priority) of the specific software project, which are offered to members of the project team as assistance in choosing measures to reduce or eliminate risks.

Stage 3. Risks planning:

- Risks reduction or elimination measures – a set of potential risks reduction or elimination measures PMR = $\{pmr_1, \dots, pmr_{19}\}$, where pmr_1 – prior training of project team members; pmr_2 – coordination of a detailed list of requirements with the customer; pmr_3 – inclusion of the agreed list of requirements of the customer in the contract; pmr_4 – exact compliance with the customer's requirements from the agreed list of requirements; pmr_5 – preliminary market research; pmr_6 – expert evaluation of the project by an experienced third-party consultant; pmr_7 – consultations of an experienced third-party consultant; pmr_8 – training to learn the necessary development tools; pmr_9 – concluding an insurance contract; pmr_{10} – use of "template" solutions from successful previous projects in project management; pmr_{11} – preparation of documents showing the importance of this project to achieve the financial goals of the developer's company; pmr_{12} – reorganization of the project team so that the responsibilities and work of team members overlap; pmr_{13} – purchase (order) of part of the components of the developed software; pmr_{14} – replacement of potentially defective components of the developed software with purchased components that guarantee the quality of work; pmr_{15} – acquisition of a more productive database(s); pmr_{16} – use of the source code generator; pmr_{17} – reorganization of the project team depending on the level of complexity of tasks and professional levels of developers; pmr_{18} – reuse of suitable software components that have been developed for other projects; pmr_{19} – analysis of the feasibility of creating this software.

The rules for determining the measures to reduce or eliminate the risks of a particular software project and the forming the set PMRER of measures for a particular software project (one, the most appropriate, measure for each risk!) are as follows:

if the risk rsp_g can be reduced or eliminated by the measure pmr_1, then $pmr_1 \in$ PMRER;

if the risk rsp_g can be reduced or eliminated by the measure pmr_2, then $pmr_2 \in$ PMRER;

…

if the risk rsp_g can be reduced or eliminated by the measure pmr_{19}, then $pmr_{19} \in$ PMRER.

Stage 4. Risks monitoring:

- Risk assessments - all risk-related values are not constant in the project. The probability of a risk event and potential losses may increase and decrease as a result of risk mitigation or elimination measures. Therefore, estimates of the probability, damage and magnitude of risk after the application of such measures are required. For each risk from a set RSP, the development team must determine the probability (in the range [0;1]) of its occurrence after the application of the chosen measure to reduce or eliminate risks. The set of probabilities of risks after the application of measures has the form: PRA=$\{pra_1,\ldots,pra_k\}$, where k is the number of risks of a particular software project. For each risk from the set RSP, the development team must determine the amount of possible losses (in the range [0;1]) from its occurrence after the application of the selected measure to reduce or eliminate risks. The set of risk losses after the application of measures is as follows: LRA=$\{lra_1,\ldots,lra_k\}$, where k is the number of risks of a particular software project. For each risk from the set RSP, it's necessary to determine its magnitude after applying the selected measure to reduce or eliminate risks. The set of risk magnitudes after the measures has the form: MRA=$\{mra_1,\ldots,mra_k\}$, where k is the number of risks of a particular software project, $mra_i = pra_i \cdot lra_i$.

3. Results & Discussion

For example, let's consider a project to develop software for job search and recruitment.

Stage 1. Risks identification. The analysis of the software project documentation showed that it lacks a description of quality and reliability characteristics, time performance characteristics, recommendations for future software maintenance, properties and possibilities of flexibility to change plans, project strategy and planning, project success forecasting. Then, according to the rules for determining the sources of risk: psr_2 =1, psr_3 =1, psr_5 =1, psr_6 =1, psr_{11} =1, psr_{14} =1, psr_{15} =1, psr_{18} =1, and the set PSR = {0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1}. Since psr_2 =1, psr_3 =1, psr_5 =1, psr_6 =1, there are technical risks. Since psr_{11} =1, there are plan risks. Since psr_{14} =1, psr_{15} =1, psr_{18} =1, there are risks of project management processes and procedures.

In addition, the analysis of the software project documentation showed that there may be reluctance of developers to use lifecycle support software, insufficient database(s) performance, defects and limited functionality of reusable software components, defective system components, changes and schedule violations work, underestimation of project time, low morale of staff, weak interaction between members of the development team, passivity and lack of competence of the project manager, inability to organize the necessary staff training, underestimation of the required number of developers, unrealistic forecasting of project results. Then, according to the rules for determining the risks for a particular program project: pre_3 = «reluctance of developers to use lifecycle support software tools», pre_6 = «insufficient performance of database(s)», pre_7 = «reusable software components have defects and limited functionality», pre_{11} = «defective system components», pre_{18} = «changes in the work schedule», pre_{19} = «violation of the work schedule», pre_{22} = «underestimation of project time», pre_{28} = «low morale of staff», pre_{29} = «weak interaction between members of the development team», pre_{30} = «passivity of the project manager», pre_{31} = «insufficient competence of the project manager», pre_{36} = «inability to organize the necessary staff training», pre_{38} = «underestimation of the required number of developers», pre_{42} = «unrealistic forecasting of project results», and the set PRE = {0, 0, «reluctance of developers to use lifecycle support software tools», 0, 0, «insufficient performance of database(s)», «reusable software components have defects and limited functionality», 0, 0, 0, «defective system components», 0, 0, 0, 0, 0, 0, «changes in the work schedule», «violation of the work schedule», 0, 0, «underestimation of project time», 0, 0, 0, 0, 0, «low morale of staff», «weak interaction between members of the development team», «passivity of the project manager», «insufficient competence of the project manager», 0, 0, 0, 0, «inability to organize the necessary staff training», 0, «underestimation of the required number of developers», 0, 0, 0, «unrealistic forecasting of project results», 0}. Since pre_3 =1, pre_6 =1, pre_7 =1, pre_{11} =1, there are technical risks. Since pre_{18} =1, pre_{19} =1, pre_{22} =1, there are plan risks. Since pre_{28} =1, pre_{29} =1, pre_{30} =1, pre_{31} =1, pre_{36} =1, pre_{38} =1, pre_{42} =1, there are risks of project management processes and procedures. According to the rules for forming the set of risks of a particular software project, the set RSP = {«reluctance of developers to use lifecycle support software tools», «insufficient performance of database(s)», «reusable software components have defects and limited functionality», «defective system components», «changes in the work schedule», «violation of the work schedule», «underestimation of project time», «low morale of staff», «weak interaction between members of the development team», «passivity of the project manager», «insufficient competence of the project manager», «inability to organize the necessary staff training», «underestimation of the required number of developers», «unrealistic forecasting of project results»}, k =14.

Stage 2. Risks analysis. For each risk from the set RSP, the development team identified the probability of its occurrence in the range [0;1]. The set of probabilities of risks has the form: PR={0.53, 0.71, 0.12, 0.15, 0.05, 0.13, 0.29, 0.41, 0.89, 0.76, 0.67, 0.91, 0.47, 0.03}. According to the rules for classifying risks according to their probabilities, we find that there are 2 risks with very low probability, 3 risks with low probability, 3 risks with medium probability, 3 risks with high probability, 3 risks with very high probability.

For each risk from the set RSP, the development team identified the amount of possible losses from its occurrence – in the range [0;1]. The set of risk losses has the form: LR={0.1, 0.5, 0.6, 0.3, 0.9, 0.5, 0.41, 0.96, 0.87, 0.76, 0.73, 0.74, 0.93, 0.94}.

For each risk from the set RSP it was determined its magnitude. The set of risk magnitudes has the form: MR={0.053, 0.355, 0.072, 0.045, 0.045, 0.065, 0.1189, 0.3936, 0.7743, 0.5776, 0.4891, 0.6734, 0.4371, 0.0282}.

Let's find the maximal mr_max and minimal mr_min elements of the set MR: mr_max =0.7743, mr_min =0.0282. Let's divide the resulting interval [0.0282; 0.7743] at three intervals: $[0.0282; 0.2769)$, $[0.2769; 0.5256)$, $[0.5256; 0.7743]$. According to the rules for identifying the level of priority of risks, we identify the level of priority and rank risks by priority: risk rsp_1 has a low level of priority; risk rsp_2 has a medium level of priority; risk rsp_3 has a low level of priority; risk rsp_4 has a low level of priority; risk rsp_5 has a low level of priority; risk rsp_6 has a low level of priority; risk rsp_7 has a low level of priority; risk rsp_8 has a medium level of priority; risk rsp_9 has a high level of priority; risk rsp_{10} has a high level of priority; risk rsp_{11} has a medium level of priority; risk rsp_{12} has a high level of priority; risk rsp_{13} has a medium level of priority; risk rsp_{14} has a low level of priority. In this case, the set of priority risks (with a high level of priority) consists of risks rsp_9, rsp_{10}, rsp_{12}; the set of secondary risks (with a medium level of priority) consists of risks rsp_2, rsp_8, rsp_{11}, rsp_{13} and the set of least risks (with a low level of priority) of a specific software project consists of risks rsp_1, rsp_3, rsp_4, rsp_5, rsp_6, rsp_7, rsp_{14}.

Stage 3. Risks planning. According to the rules for determining the measures to reduce or eliminate the risks of a particular software project and the forming the set of measures for a particular software project, the set PMRER ={« training to learn the necessary development tools», «acquisition of a more productive database(s)», «reuse of suitable software components that have been developed for other projects», «replacement of potentially defective components of the developed software with purchased components that guarantee the quality of work», «consultations of an experienced third-party consultant», «exact compliance with the customer's requirements from the agreed list of requirements», «expert evaluation of the project by an experienced third-party consultant», «prior training of project team members», «reorganization of the project team depending on the level of complexity of tasks and professional levels of developers», «reorganization of the project team depending on the level of complexity of tasks and professional levels of developers», «reorganization of the project team so that the responsibilities and work of team members overlap», «use of "template" solutions from successful previous projects in project management», «consultations of an experienced third-party consultant», «consultations of an experienced third-party consultant»}.

Stage 4. Risks monitoring. For each risk from the set RSP, the development team determined the probability (in the range [0;1]) of its occurrence after the application of the selected measures to reduce or eliminate risks. The set of probabilities of risks after the application of measures is as follows: PRA={0.21, 0.1, 0.02, 0.02, 0.02, 0.03, 0.08, 0.1, 0.19, 0.14, 0.05, 0.41, 0.27, 0.01}. For each risk from the set RSP, the team of developers identified the amount of possible losses (in the range [0;1]) from its occurrence after the application of the selected measures to reduce or eliminate risks. The set of risk losses after the application of measures is as follows: LRA={0.1, 0.5, 0.2, 0.05, 0.9, 0.5, 0.41, 0.96, 0.1, 0.2, 0.1, 0.54, 0.93, 0.94}. For each risk from the set RSP, it was determines its magnitude after applying the selected measures to reduce or eliminate risks. The set of risk magnitudes after measures is as follows: MRA={0.021, 0.05, 0.04, 0.001, 0.018, 0.015, 0.0328, 0.096, 0.019, 0.028, 0.005, 0.2214, 0.2511, 0.0094}. Comparison of the sets MR and MRA allows us to conclude that after the application of selected measures to reduce or eliminate risks, the magnitude of risks has decreased significantly – Figure 1.

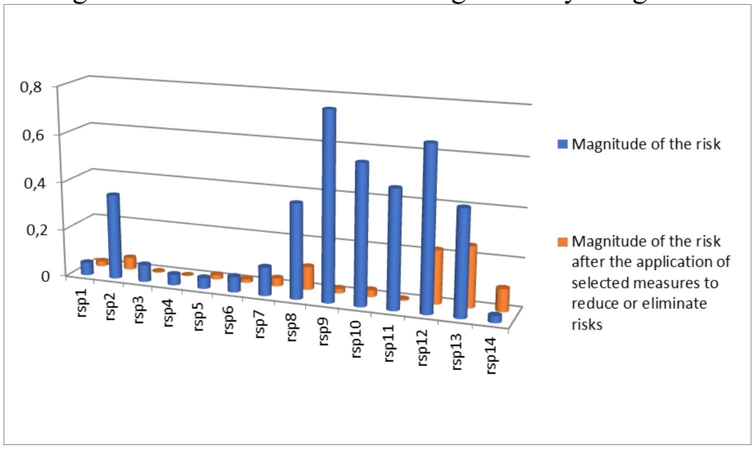

Figure 1: Magnitudes of the risks before and after the application of selected measures to reduce or eliminate risks

Therefore, the proposed method of the software risks management makes it possible to identify sources of risk and possible risks for any software project, as well as to assess risks, determine their priority and measures to reduce or eliminate risks. In addition, the method allows risks assessment after the application of selected measures to reduce or eliminate risks, which makes it possible to select the best measure to minimize the magnitude of each risk. The presented method provides a mathematical basis for a risks management process, which reduces the complexity and increases the effectiveness of risk management.

4. Conclusions

The result of any project depends on the number and magnitude of risks of insufficient software functionality, non-compliance with project deadlines, budget overruns. The task of developers is to reduce and eliminate risks. Reducing the risks of a software project helps to increase its success, quality, efficiency and effectiveness. Therefore, risks management should be one of the foundations of project management, and the actual task now is to improve risk management in software development.

From the results of the analysis of the current state of the software development industry it follows that a promising area of research is the development of a mathematical basis or a method of risks management in software development. Therefore, the main task of this study is detailing and formalizing the method of the software risks management..

The paper proposes a method of software risks management, which allows identifying sources of risks and possible risks for any software project, as well as to assess risks, determining their priority and measures to reduce or eliminate risks. In addition, the method allows risks assessment after the application of selected measures to reduce or eliminate risks, which makes it possible to select the best measure to minimize the magnitude of each risk. The conducted experiment allows us to conclude that after the application of selected measures to reduce or eliminate risks, the magnitude of risks has decreased significantly. Herewith, the presented method provides a mathematical basis for a risks management process, which reduces the complexity and increases the effectiveness of risks management.

The prospect for further research by the authors is to develop a software risks management system, which will be based on the proposed in the paper method of the software risks management.

References

[1] Latest study shows rise in project failures, 2019. URL: http://kinzz.com/resources/articles/91-project-failures-rise-study-shows.

[2] H. Shane, W. Stéphane, Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch, 2015. URL: http://www.infoq.com/articles/standish-chaos-2015.

[3] The Standish Group Report CHAOS, 2014. URL: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

[4] PMI's Pulse of the Profession 9-th Global Project Management Survey, 2017. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

[5] A Look at 25 Years of Software Projects. What Can We Learn?, 2017. URL: https://speedandfunction.com/look-25-years-software-projects-can-learn/.

[6] M. Bloch, S. Blumberg, J. Laartz, Delivering large-scale IT projects on time, on budget, and on value, 2012. URL: http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value.

[7] T. Hovorushchenko, O. Pavlova, Method of Activity of Ontology-Based Intelligent Agent for Evaluating the Initial Stages of the Software Lifecycle. Advances in Intelligent Systems and Computing 836 (2019) 169-178. doi:10.1007/978-3-319-97885-7_17.

[8] O. Pomorova, T. Hovorushchenko, The Way to Detection of Software Emergent Properties, in: Proceedings of the 2015 IEEE 8-th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2015, Warsaw, 2015, vol. 2, pp. 779-784. doi: 10.1109/IDAACS.2015.7341409.

[9] O. Pomorova, T. Hovorushchenko, Artificial neural network for software quality evaluation based on the metric analysis, in: Proceedings of IEEE East-West Design & Test Symposium, EWDTS'2013, Kharkiv, 2013, pp. 200-203. doi: 10.1109/EWDTS.2013.6673193.

[10] I. Izonin, R. Tkachenko, N. Kryvinska, K. Zub, O. Mishchuk, T. Lisovych, Recovery of Incomplete IoT Sensed Data using High-Performance Extended-Input Neural-Like Structure. Procedia Computer Science 160 (2019) 521–526. doi: 10.1016/j.procs.2019.11.054

[11] R. Tkachenko, I. Izonin, N. Kryvinska, V. Chopyak, N. Lotoshynska, D. Danylyuk, Piecewise-linear Approach for Medical Insurance Costs Prediction using SGTM Neural-Like Structure. CEUR-WS 2255 (2018) 170–179.

[12] T. Huckle, T. Neckel, Bits and Bugs: A Scientific and Historical Review of Software Failures in Computational Science. Society for Industrial & Applied Mathematics, Florida, 2019.

[13] C. Hobbs, Embedded Software Development for Safety-Critical Systems, CRC Press, Taylor & Francis Group, Orlando, 2016.

[14] I. Sommerville, Engineering Software Products: An Introduction to Modern Software Engineering, Pearson, London, 2019.

[15] S. Wagner, Optimising Analytical Software Quality Assurance. Software Quality: Quality Intelligence in Software and Systems Engineering, in: Proceedings of the 12th International Conference on Software Quality: Proceedings, Vienna, 2020, pp. 134-138. doi: 10.1007/978-3-030-35510-4_9.

[16] G. O'Regan, Concise Guide to Software Engineering: From Fundamentals to Application Methods (Undergraduate Topics in Computer Science), Springer International Publishing, Switzerland, 2014. doi: 10.1007/978-3-319-57750-0.

[17] O. Drozd, K. Zashcholkin, R. Shaporin, J. Drozd, Y. Sulima, Development of ICT Models in Area of Safety Education, in: Proceedings of IEEE East-West Design & Test Symposium, EWDTS'2020, Varna, 2020, pp. 212–217. doi: 10.1109/EWDTS50664.2020.9224861.

[18] A. Drozd, V. Kharchenko, S. Antoshchuk, J. Sulima, M. Drozd, Checkability of the digital components in safety-critical systems: problems and solutions, in: Proceedings of IEEE East-West Design & Test Symposium, EWDTS'2011, Sevastopol, 2011, pp. 411–416. doi: 10.1109/EWDTS.2011.6116606.

[19] O. Drozd, K. Zashcholkin, O. Martynyuk, O. Ivanova, J. Drozd, Development of Checkability in FPGA Components of Safety-Related Systems. CEUR-WS 2762 (2020) 30-42.

[20] R. Natella, D. Cotroneo, H. Madeira, Assessing Dependability with Software Fault Injection: A Survey. ACM Computing Surveys 48 (2016) 1-55. doi: 10.1145/2841425.

[21] M. Kabir, M. Rehman, S. Majumdar, An analytical and comparative study of software usability quality factors, in: Proceedings of the 7th IEEE International Conference on Software Engineering and Service Science, ICSESS'2016, Beijing, 2016, pp. 800-803. doi: 10.1109/ICSESS.2016.7883188.

[22] I. Margarido, J. Faria, R. Vidal, M. Vieira, Classification of Defect Types in Requirements Specifications: Literature Review, Proposal and Assessment, in: Proceedings of the 6th Iberian Conference on Information Systems and Technologies, CISTI' 2011, Chaves, 2011, pp. 1-6.

[23] T. Hovorushchenko, O. Pavlova, Evaluating the software requirements specifications using ontology-based intelligent agent, in: Proceedings of 2018 IEEE International Scientific and Technical Conference "Computer Science and Information Technologies", CSIT'2018, Lviv, 2018, vol.1, pp.215-218. doi: 10.1109/STC-CSIT.2018.8526730.

[24] T. Hovorushchenko, O. Pavlova, M. Bodnar, Development of an intelligent agent for analysis of nonfunctional characteristics in specifications of software requirements. Eastern-European Journal of Enterprise Technologies 1 2 (2019) 6-17. doi: 10.15587/1729-4061.2019.154074.

[25] K. Naik, P. Tripathy, Software Testing and Quality Assurance: Theory and Practice, Wiley Publishing, New York, 2011.

[26] G. Blokdyk, Nintendo Software Planning & Development, CreateSpace Independent Publishing Platform, London, 2018.

[27] J. Capers, Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies, McGraw-Hill Education, Ireland, 2010.