Intelligent team work planning: a model for increasing sprint value

Olha Yanholenko [1,†], Marina Grinchenko [1,*,†], Mykyta Rohovyi [1,†], Olena Yakovleva [2,†], and Anton Rogovyi [1,†]

[1]National Technical University "Kharkiv Polytechnic Institute", Kyrpychova str. 2, Kharkiv, 61002, Ukraine
[2]Bratislava University of Economics and Management, Furdekova 16, Bratislava, 85104, the Slovak republic,

Abstract

Agile software development methodologies are currently the dominant approach to IT project management. However, sprint planning and decision-making processes within Agile teams still largely rely on manual input and intuitive judgments, often leading to suboptimal resource allocation and unmanaged risks. A critical factor in successful project execution is the quality of requirements and tasks in the backlog. Tasks formulated in natural language frequently contain ambiguities or inconsistencies, complicating their interpretation and the team's ability to plan effectively. Such defects in task descriptions can result in misunderstandings, increasing the likelihood of errors and product defects. An analysis of existing approaches reveals the absence of an integrated model that simultaneously considers task formulation quality, planning mechanisms, and personalized task assignment. This paper introduces an intelligent sprint planning model that formalizes the selection of tasks based on their characteristics, team member preferences, business value, and associated risks. The decision-making process is supported by machine learning algorithms and large language models. Experimental evaluation of the proposed model on benchmark datasets confirmed its sensitivity to task description quality: reducing the clarity of just two task descriptions increased the aggregated defect risk by 50% and decreased the integral sprint value by 10–15%. In contrast, the use of stable task assignment preserved sprint value under similar conditions. Therefore, the proposed approach enables the enhancement of sprint outcomes by incorporating task text quality and defect risk into the planning process.

Keywords.

AGILE, project, risk, system model; project management, decision support, natural language processing, project team, method, task description

1. Introduction

Agile software development methodologies [1] have now become the standard for managing IT projects. At the same time, the process of sprint planning and decision-making in Agile teams is still largely manual and based on the experience and subjective opinions of team members [2]. This causes a number of problems: in particular, reliance on intuitive estimates can lead to ineffective risk management and suboptimal resource allocation in the project. The quality of the requirements and tasks (user stories) in the backlog also significantly affects the success of the project. Requirements described in natural language often contain ambiguities or contradictions, making it difficult for the team to understand the tasks and plan them. Such defects in task descriptions can lead to misinterpretation of requirements, which increases the risk of errors and defects in the product. Additionally, the tasks distribution is still relevant: improper or uneven distribution leads to overloading of individuals and a decrease in the quality of teamwork. Initial failures in team

ISW-2025: Intelligent Systems Workshop at 9th International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2025), May 15–16, 2025, Kharkiv, Ukraine

* Corresponding author.
† These authors contributed equally.
olha.yanholenko@khpi.edu.ua (O. Yanholenko); marina.grynchenko@khpi.edu.ua (M. Grinchenko); nikrogovoy@gmail.com (M. Rohovyi); olena.yakovleva@vsemba.sk (O. Yakovleva); anton.rogovyi@khpi.edu.ua (A. Rohovyi)
0000-0001-7755-1255 (O. Yanholenko); 0000-0002-8383-2675 (M. Grinchenko); 0000-0002-7902-3592 (M. Rohovyi); 0000-0002-6129-6146 (O. Yakovleva); 0000-0002-8178-4585 (A. Rohovyi)

productivity are often caused by the exhaustion of overworked developers and irrational distribution of tasks, which negatively affects the quality of the product.

The growing demand for more efficient teamwork in Agile-driven IT projects highlights the relevance of intelligent planning – an approach that relies not only on managerial intuition but also on data- and knowledge-based decision-making. Recent advances in artificial intelligence, particularly in machine learning and large language models (LLMs), offer significant opportunities for automating and enhancing the planning process. Integrating AI into Agile workflows can improve the validity and consistency of planning decisions, enabling, for example, more accurate risk identification and optimized resource allocation through AI-powered decision support systems. In particular, the use of natural language processing (NLP) tools based on LLMs allows for the automatic analysis and refinement of task descriptions, improving their clarity and comprehensibility for the development team. Additionally, the application of stable matching algorithms to task assignment makes it possible to consider both task requirements and individual preferences of team members, ensuring more balanced and satisfactory task distribution.

Initial research by the authors has already begun to explore these directions [3], including studies on using NLP [4, 5]. Building upon these foundations, the scientific challenge is to develop comprehensive models and methods for intelligent planning of team work in Agile IT projects. Addressing this challenge will lead to improved planning accuracy, enhanced team productivity, and reduced risks and defects through better task formulation and allocation powered by artificial intelligence.

This study introduces a novel intelligent planning model for Agile sprint management and investigates its structure, key parameters, and practical application. The following research questions are formulated:

- RQ1: How does the use of AI components (LLMs, NLP, stable matching) influence planning quality and team satisfaction?
- RQ2: How can an intelligent planning model be designed to support task formulation and distribution in Agile sprint planning?
- RQ3: What parameters of the model have the most significant impact on sprint value and team performance?

2. Related Work

Research on improving project team performance under risk can be broadly divided into two main directions: [1] studies that analyze the impact of task descriptions on team effectiveness and [2] works focused on the evaluation of risk management scenarios in project execution.

Let's consider studies related to the impact of task descriptions on team performance. Several works examine how the quality and structure of task descriptions affect team outcomes. Risk factors in software projects are classified in [6], though without analyzing their interdependencies or contextual factors. A method for relative task evaluation in Agile settings is proposed in [7], reducing estimation time and skill requirements. For Kanban teams, [8] introduces a time estimation approach accounting for dynamic task pool changes and team productivity.

The quality of requirements descriptions is a separate line of study. The notion of “requirement smells” is introduced in [9] to detect early-stage defects, while [10] applies semi-supervised categorization to non-functional requirements for recommender systems. Linguistic classification is used in [11] to reduce ambiguity in user stories, improving clarity in adaptive development contexts.

Text classification and vectorization approaches are used widely [12, 13] and successfully applied to project management processes [14, 15]. These include TF-IDF with SVM for IT help desk tasks [14], semantic analysis of terminological ambiguity [16], and visual analysis tools like DeepNLPVis [13] to understand model behavior. Task descriptions are also used to predict problem-solving success in [17, 18].

Advanced NLP and deep learning techniques are explored in [19, 20], where models like RoBERTa and CodeBERT-MLM are applied to software defect prediction. Visualization and classification methods using convolutional neural networks are also discussed.

Several tools and approaches aim to automate analysis and detection of defects in requirements. These include the TABASCO tool for detecting ambiguities [21], syntax pattern recognition via NLP and clustering [22], and classification/prioritization of requirement issues using ML models like LR, SVM, and KNN [23].

Evaluation of risk management scenarios in project execution is considered by number of researchers. Risk prioritization strategies for software development are proposed in [24], while [25] presents a structural equation model linking key risk factors to project outcomes based on a survey of 145 projects.

Uncertainty management is addressed in [26], which outlines five methods and 18 practices to reduce it, and [27], which presents a simulation model linking risk, cost, and planning. Multi-agent approaches, including Q-learning, are used in [28] for adaptive risk response, though the model does not handle multiple concurrent risks.

Machine learning for predicting project management failures is discussed in [29], identifying the SVM algorithm as particularly effective. System dynamics modeling for project planning and change management is developed in [30], accounting for complex feedback loops.

The analysis of existing research reveals that numerous approaches utilize natural language processing, machine learning, and semantic analysis to classify and refine task descriptions, which directly impacts team performance and sprint outcomes. However, despite the diversity of tools and methods, current research lacks a unified approach that combines task formulation quality, risk-aware planning, and personalized task distribution into a coherent framework suitable for Agile team environments. This gap underlines the need for development of a model of intelligent sprint planning, which incorporates the properties of tasks, team preferences, and the potential impact of task quality and associated risks.

3. Materials and Methods

3.1. Intelligent Team Work Planning Framework

The developed in [4] reference model of project team management reflects the relationship of four models: a model of project team behavior, a model for assessing the quality of sprint task formation, a model for determining the distribution of tasks in the project team, and a model for generating recommendations. This model of project team management sets the basis for all the entities and information flows that need to be taken into account to formulate recommendations for the sprint.

The next step is to detail how these flows are transformed into decisions. For this purpose, we built the intelligent planning framework it is shown in Fig. 1., which implements a step-by-step approach to automating and optimizing the sprint planning process based on the use of estimation, forecasting, and recommendation models. This framework integrates artificial intelligence models with Agile management practices and specifies how the recommendation model transforms the collected data into practical solutions. Let's consider the elements of the developed framework in detail.

1. Assessment of the business value of the task. This component allows you to prioritize the task in terms of its contribution to the achievement of the project's business goals. The results of this assessment are used both in the model for selecting tasks for the sprint backlog and in the model for distributing tasks among performers.

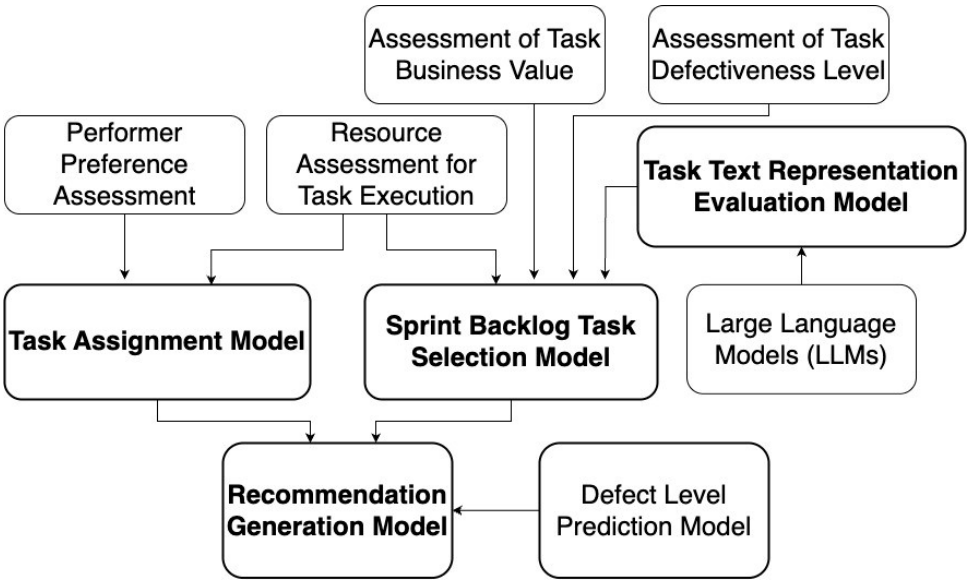

Figure 1: Framework for intelligent work planning of an IT project team.

1. Estimating resources for tasks. This module involves forecasting the labor costs and resources required to complete each task. This information is critical for planning the team's workload and is used to select tasks for sprinting.
2. Estimating the defect rate of tasks. It involves using historical data to assess the potential complexity and risk of defects as a result of the task implementation. This parameter influences decision-making in the task selection model.
3. Estimating the defect rate of tasks. It involves using historical data to assess the potential complexity and risk of defects as a result of the task implementation. This parameter influences decision-making in the task selection model.
4. Model for evaluating the textual representation of tasks which is presented in [3]. This model, based on LLM, automatically analyzes the task description to identify incompleteness, ambiguity, or poor quality of the wording. The quality of the task formulation, in turn, affects the probability of defects - the lower the quality, the higher the probability of defects.
5. The model for selecting tasks in the sprint backlog integrates the results of assessing business value, resources, risks, and task description quality to form a balanced set of tasks in the sprint.
6. The model of task assignment between performers uses the results of previous estimates to effectively assign tasks to individual team members, taking into account their competence, workload, and specialization. The proposed model is considered in [5].
7. The defect rate prediction model is based on task evaluation data and the quality of their description to predict the probability of errors that may occur during implementation.
8. The recommendation generation model is an integration component that summarizes data from all models and generates final recommendations for the project manager or team on the sprint plan, including a list of tasks, distribution of performers, and risks.

The proposed framework demonstrates the possibility of using intelligent models to improve the sprint planning process in an IT team. It combines six specialized analytical blocks into a single model for intelligent sprint planning. At the end of the process, the defect prediction model checks the plan, adjusting it if the risk exceeds the acceptable threshold, while the recommendation model combines the results of all subsystems into a set of actions that the manager can understand. In this way, the framework integrates priorities (business value), task content assessments (text quality, defectiveness, labor intensity), and generates reasonable recommendations for both sprint content and task distribution within the team. The integration of task estimation, resource planning,

language models, and defect prediction allows for informed decision-making support, increasing teamwork efficiency and software product quality.

3.2. A Method for Intelligent Team Work Planning

As mentioned earlier, in the context of the Agile methodology and sprint approach, the process of forming an optimal set of tasks for the sprint, as well as their balanced distribution among team members, is critical. Successful completion of this task requires consideration of numerous factors: business value of tasks, resource intensity, inter-task dependencies, quality of task description, and compliance of tasks with the competencies of performers.

To build a formal model of intelligent planning of the IT project team, it is necessary to clearly define the sets of tasks and performers, as well as to introduce the main parameters that characterize these tasks and affect the process of their selection and distribution. In particular, each task should be evaluated according to a number of criteria, such as business value, labor intensity, complexity, sequence of execution, clarity of formulation, and compliance with the preferences of the performers.

Let:

- $T = \{t_1, t_2, \dots, t_N\}$ - a set of tasks from the project backlog;
- $D = \{d_1, d_2, \dots, d_M\}$ - a set of developers (project team).

Each task $t_i \in T$ is characterized by an assessment of the business value $V(t_i)$ for the project as a whole, and the requirements $C(t_i)$ for its implementation (complexity, performer's qualifications, deadline). For each developer $d_j \in D$, their qualifications $Q(d_j)$ are determined, which can be compared with the requirements of the sprint tasks. Each developer $d_j \in D$ has its own preferences for backlog tasks $P(d_j, t_i)$.

Let $E(t_i), \forall\, i = \overline{1,N}$ defines the required resources, including the execution time, and let $A(d_j), \forall j = \overline{1,M}$ denotes the available working time of each developer $d_j \in D$. Let us denote the total available resources of the team per sprint by E^S, which in general can take into account more complex dependencies than the total available time.

Taking into account the previously discussed risk factors of task failure and defectiveness, consider the assessments $U(t_i), t_i \in T$ which are reflected the clarity of the task description.

It should be noted that some tasks have interdependencies. Let's take into account the dependence on the sequence of tasks. Let's denote $G(t_k, t_i), t_k, t_i \in T$ as an indicator of the sequence of tasks in such a way that task t_k must be completed before t_i. Let denote that $H^-(t_i)$ indicates dependencies, i.e. a set of tasks that must be completed before starting t_i, and $H^+(t_i)$ - the set of tasks that depend on the task t_i.

Then the formal task statement can be formulated:

- Find a subset of tasks $T' \subseteq T$ that will be selected for execution in the sprint such that the total labor intensity of $E(T')$ does not exceed the team's resources, and the tasks T' take into account dependencies and are ranked by clarity, importance, and risks.
- Find a bijection $f: T' \to D$ that allows you to distribute tasks among performers based on preferences and ensures a sustainable distribution. Sustainable allocation means that there are no alternative reallocations that would be more favorable for performers or tasks.

The first step is to select a subset of tasks from the overall project backlog that should be included in the sprint. This choice should take into account the team's resource constraints (primarily the total labor intensity of the tasks), as well as the importance of the tasks in terms of business goals, complexity, dependencies between tasks (i.e., the sequence of their implementation), clarity of the technical description, and risks associated with their implementation.

The second step is to assign the selected tasks to specific team members in a way that maximizes their professional competencies, individual preferences for task types, and ensures an even and efficient workload. At the same time, the distribution of tasks should be stable and conflict-free, and the assignment should increase the motivation and satisfaction of team members.

Let's consider a formalized description of the method of intelligent planning of the IT project team.

Stage 1. Evaluation and ranking of tasks. The goal is to prioritize the tasks in the backlog.

The total risk of the task:

$$R(t_i) = \gamma_1\big(1 - U(t_i)\big)V(t_i) + \gamma_2 \sum_{t_k \in H^+(t_i)} (1 - U(t_k)), i = 1..N$$

Takes into account the clarity of the formulation (quality) and importance of the task, as well as dependencies. Priority of the task:

$$P(t_i) = \alpha V(t_i) + \beta C(t_i) - R(t_i), i = 1..N.$$

Stage 2. Task selection (filtering). The goal is to select a subset of$T' \subseteq T$ that is relevant:

$$\max_{T' \subseteq T} \sum_{t_i \in B'} P(t_i), i = 1..N$$

$$\sum_{t_i \in T'} E(t_i) \le \sum_{d_j \in D} A\big(d_j\big), j = 1..M, i = 1..N$$

Stage 3. Task distribution. The goal is to assign tasks to performers

$$max \sum_{t_i \in T'} W(t_i, f(t_i)), i = 1..N.$$

where:

- $W\big(t_i, d_j\big)$ – is a satisfaction function (task and developer matching);
- $i = 1..N, j = 1..M.$

$$f_i \quad T' \to \mathrm{D}, \qquad f(t_i) = d_j, \qquad S(d_j) \subseteq S(t_i)$$

$$\sum_{d_j \in f^{-1}(t_i)} E(t_i) \le A(t_i) \quad \forall t_i \in T, i = 1..N, j = 1..M$$

Thus, the proposed method of intelligent planning of the IT project team's work is a comprehensive approach that ensures informed decision-making at all stages of sprint formation - from task selection to task distribution among performers. Its key advantage is the ability to integrate heterogeneous factors such as business value, technical complexity, inter-task dependencies, clarity of formulation, and human preferences into a single evaluation and optimization system.

3.3. Model of the task of intelligent team planning

For the purposes of this study, sprint value is an aggregate (business) metric of the expected effect of the planned sprint tasks. Risks are defined as a decrease in value caused by certain factors (unclear descriptions, defects, team overload, etc.).

The goal of the model is to build a sprint plan that minimizes the probability of defects while maximizing business value with limited team resources.

Let:

- $x_i \in \{0,1\}, i = \overline{1,N}$ is a sign of selecting the taskt_i for execution in sprint;

- $y_{ij} \in \{0,1\}, j = \overline{1,M}, i = \overline{1,N}$ is a sign of assigning a taskt_i to a developer d_j.

For each pair $(t_i, d_j), t_i \in T, d_j \in D$, we assume the following estimates to be certain:

- $DS(d_j, t_i)$ is the satisfaction of the developer d_j with the assignment of the task t_i , which takes into account the preferences of the performer;
- $CS(t_i, d_j)$ is the consistency of assignment of task t_i to developer d_j, which takes into account the performer's compliance with the task requirements.

We assume that the consistency of the task assignment and the satisfaction of the contractor assigned to this task increases its weight in the sprint, as it positively affects the project results as a whole.

High requirements for the task (complexity) add value to it and the sprint as a whole. Therefore, given the goal of increasing the value of the sprint, the value of the task $t_i \in T$ can be written as follows:

$$V_{total}(t_i) = \alpha * C(t_i) * V(t_i) * \left(\sum_{j=1}^{M} (CS(t_i, d_j) + DS(d_j, t_i)) * y_{ij} \right) * x_i,$$

where:

- α is scaling factor.

The task selected for the sprint is critical to the results of the sprint and the project as a whole.

Taking into account the previously discussed risk factors, consider $I(t_i, d_j), t_i \in T, d_j \in D$ as an indicator of the condition that there is a better allocation for the task t_i (there is a performer who better meets the task requirements than d_j) and there is a more attractive task for the developer d_j than t_i. We will call this condition a blocking condition, and the indicator $I(t_i, d_j)$ will be called a blocking indicator. This indicator reflects the risk that the task will not be completed properly or that the performer will switch to another task that is more interesting to him.

Then the value of the task increases with the tasks that depend on it, etc.:

$$\widetilde{V_{total}}(t_i) = \alpha * C(t_i) * V(t_i) * \left(\sum_{j=1}^{M} (CS(t_i, d_j) + DS(d_j, t_i)) * y_{ij} \right) * x_i + \left(\sum_{k=1}^{N} (V_{total}(t_k) + D(t_i, t_k)) \right) * x_i, \forall t_i \in T$$

Thus, the value of the task $\widetilde{V_{total}}(t_i)$ takes into account the value of all tasks t_k that depend on the execution of the task $.t_i$

Then the criterion for maximizing the total value of sprint tasks can be written as follows:

$$V^S = \sum_{i=1}^{N} \widetilde{V_{total}}(t_i) \to max$$

The value of a sprint is defined as the total value of the tasks selected for the sprint, taking into account the risks of their defectiveness, etc.:

$$R(t_i) = \beta * \left(1 - U(t_i)\right) * V(t_i) * \left(\sum_{j=1}^{M} I(t_i, d_j) * y_{ij}\right) * x_i, \forall t_i \in T,$$

where:

- β is scaling factor.

Taking into account the dependence of tasks (sequence of execution), the risk of selecting the taskt_i for sprinting can be written as follows:

$$\tilde{R}(t_i) = \beta \left[\gamma_1 * \left(1 - U(t_i)\right) + \gamma_2 * \sum_{k=1}^{N} \left(1 - U(t_k)\right) * D(t_i, t_k) \right] * C(t_i) * \\ * \left(\sum_{j=1}^{M} I\left(t_i, d_j\right) * y_{ij} \right) * x_i, \forall t_i \in T$$

where:

- γ_1, γ_2 are the scaling factors.

Then the total risk of the tasks selected for execution in the sprint:

$$R^S = \sum_{i=1}^{N} \tilde{R}(t_i) \rightarrow min.$$

Sprint planning is the process of selecting, ordering, and distributing tasks to be performed within a limited time interval, taking into account limited resources and subject to optimizing the value of the sprint.

Sprint value (planning criterion) is a balance between the overall value of tasks and the risks associated with their execution.

Thus, the criterion for selecting tasks and distributing them among performers within the sprint can be formulated:

$$W = V^S - R^S \rightarrow max.$$

So, the task of planning the team's work during the sprint can be formulated as follows:

Find a vector of features for selecting sprint tasks:

$$X^S = (x_1, x_2, \dots, x_N).$$

The vector of assigning tasks to performers:

$$Y^S = (y_{11}, \dots, y_{1m}, y_{21}, \dots, y_{2m}, \dots, y_{nm}),$$

that maximize the value of the sprint value function

$$W = V^S - R^S$$

and satisfy the constraints.

1. The total labor intensity of the sprint tasks does not exceed the available team resource and the distributed (assigned) tasks do not exceed the resource of the corresponding performer:

$$\sum_{i=1}^{N} C(t_i) * E(t_i) * x_i \leq E^S,$$

where:

- $E(t_i)$ is the task labor intensity;
- $C(t_i)$ is the task complexity.

$$\sum_{i=1}^{N} C(t_i) * E(t_i) * y_{ij} \leq A(d_j), \forall j = \overline{1, M}.$$

2. Each performer during the sprint cannot perform more than a certain (fixed) number of tasks simultaneously:

$$\sum_{i=1}^{N} y_{ij} \leq L, \forall j = \overline{1, M}$$

Where L is a constant that determines the maximum number of tasks per sprint for one performer (usually 1-3 tasks).

3. Performing a sequence of tasks:

$$G(t_i, t_k) * x_i \geq G(t_i, t_k) * x_k, \forall i, k = \overline{1, N}.$$

Thus, the proposed model of intelligent planning of the IT project team formalizes the process of selecting tasks for a sprint, taking into account a set of critical factors that affect the effectiveness of team activities. It is based on an optimization approach to maximizing the value of the sprint, which is defined as a balance between the total utility (value) of the tasks planned for execution and the total risks that accompany their execution and form defects. The developed model includes a multicomponent function of the target cost of a task, which takes into account its business value, labor intensity, level of coordination with the performer, performer satisfaction, and the presence of blockages (as a potential risk of failure to complete the task). At the same time, the risks are assessed taking into account the vagueness of the task description, possible dependencies and conflicts, and potential defects in execution. Integration of the logical variablesx_i and y_{ij} , which are responsible for the selection of tasks and their assignment to performers, allows building formal constraints that guarantee compliance with the availability of resources, team competencies, and sprint requirements. Thus, the model allows for coordinated planning focused on achieving both business goals and improving the quality of development.

In general, the presented model is the basis for decision support in the process of sprint planning, combining mathematical rigor and adaptability to the team structure, and taking into account the qualitative and quantitative characteristics of tasks. The proposed model is the basis for building intelligent project management support systems in the IT field.

4. Results and Discussion

To test the model's performance, a control set of 4 user-stories and 3 developers was generated. Several experiments with parameter variation were planned. The initial data of the baseline variant were chosen as follows (Table 1 and Table 2). Consistency matrix is presented in Table 2, it represents the values of consistency of task assignment $CS \in [0; 1]$, where 1 - perfect correspondence of the developer's competencies to the task requirements, 0 - complete inconsistency.

Table 1
Initial data for calculation

Parameter	t_1	t_2	t_3	t_4
Business value *V(t)*	10	8	6	7
Complexity/weighting factor *C(t)*	3	2	1	2
Labor intensity (hours) *E(t)*	6	4	3	5
Clarity of description *U(t)*	0,9	0,8	0,7	0,85

Table 2
Consistency matrix *CS*

Task ID	c_1	c_2	c_3
t_1	0,9	0,7	0,6
t_2	0,8	0,8	0,5
t_3	0,6	0,7	0,4
t_4	0,7	0,6	0,9

The logical dependencies between the tasks were represented by the matrix *G*. The following was chosen:t_2 must be executed before .t_3

The smoothing parameters are defined as follows: $\alpha = \beta = 1$, $\gamma_1 = 0.6$, $\gamma_2 = 0.4$.

We consider defects to be the situation when the blocking indicator is triggered and the description clarity is $U<0.7$.

In the following experiments, we used an alternative distribution of tasks, a decrease in the clarity of the task description, and a variation of the risk weight .β.

The optimization problem was solved by the MILP solver OR-Tools. A comparison of the results is shown in Table 3.

Table 3
Comparison of calculation results

Variant	$\sum V$	$\sum R$	W	Defects
Base case: (U= {0.85,0.75,0.90,0.95}; α=β=1, γ_1=γ_2=0.5)	90,15	7,53	82,62	0
Alternative distribution*.	90,15	15,29	74,86	1
Decreased clarity of description (U={0.95,0.60,0.50,0.95})	90,15	11,85	78,3	1
Variation of risk weight (β=1.5)	90,15	11,30	78,85	1

*Assumes that each task was assigned to the developer who had the lowest total *CS + DS* score for that task in the original distribution, but at the same time met the resource constraint (time/complexity) to keep the sprint feasible.

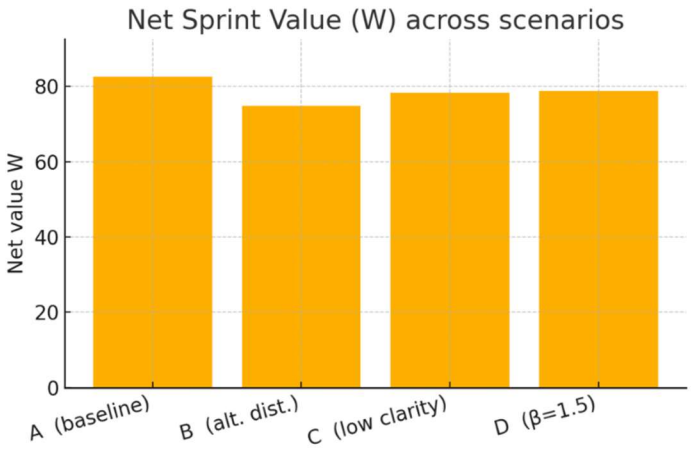

Figure 2: Comparison of the sprint's net value.

Fig. 2 clearly shows how different the outcome of the sprint can be under the four scenarios.

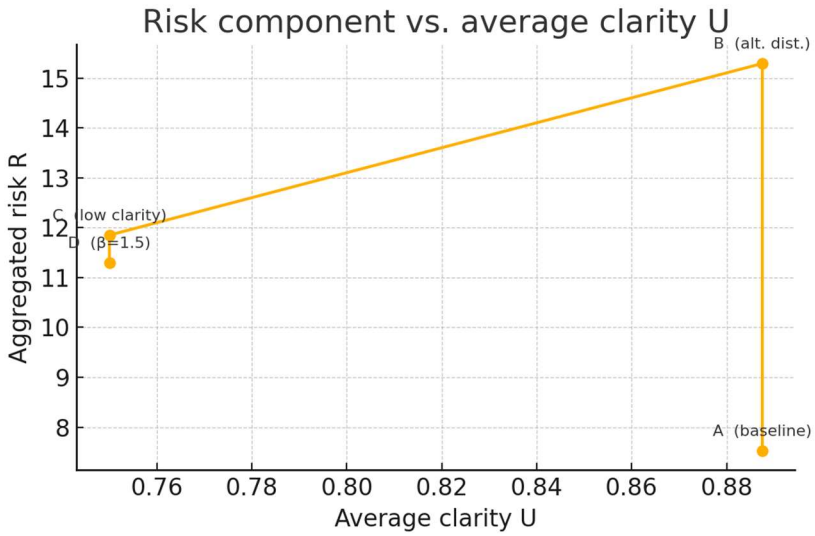

Figure 3: Dependence of total risk on the average clarity of task description *U*.

Fig. 3 clearly demonstrates that the lower the average clarity of the descriptions, the higher the aggregate risk of defects, and the growth is close to linear. For example, moving from $U\approx0.89$ to $U\approx0.75$ increases the risk by an average of one-third. An additional increase in β (plan *D*) does not increase the risk as much as a decrease in text quality, but it also has a significant impact on the result.

The analysis of the results allowed us to draw the following conclusions:

- the quality of the task description has the greatest impact on the results. Reducing $U(t_2)$, and $U(t_3)$,) by about 30% increased the risk component by 57%, and the integrated benefit of decreased more than changing any other parameter. This confirms the expediency of automatically "cleaning up" the quality of user-story before the planned sprint;
- increasing the risk weight from 1 to 1.5 resulted in a drop in the overall W score by only 1.2%, allowing the manager to gain additional reliability with virtually no loss in business value;
- reallocation performed without taking into account stable mapping (alternative allocation) increases the number of blocking pairs and almost doubles the risk of defects without adding any value.

To demonstrate the sprint scheduling model, we took an example with 5 tasks and 3 developers. The output data is presented in Tables 4 and 5.

Table 4
Task metrics

Task ID	Business value V	Complexity C	Clarity U	E (hours)	Dependence on other tasks	Required skills
t_1	9	3	0,9	6	-	{Python, SQL}
t_2	6	2	0,7	4	t_1	{SQL}
t_3	5	4	0,8	8	-	{JS, React}
t_4	8	3	0,95	4	t_1	{Python}
t_5	4	1	0,4	2	-	{HTML}

Table 5
Initial data for calculation

Developer	Skills	Available hours	Top preferences
d_1	Python, SQL	16	$t_1 \rightarrow t_4$
d_2	JS, React	16	t_3
d_3	Python, HTML	16	$t_4 \rightarrow t_5$

The model was run 2 times in the following scenarios:

- scenario (a), we use the same smoothing parameters as in the previous experiment ($\alpha = \beta = 1$, $\gamma_1 = 0.6$, $\gamma_2 = 0.4$, initial clarity U);
- scenario (b), use the same weights, but reduce the clarity of the description of tasks t_2 and t_3.

First, we calculated risks and priorities. The results for both scenarios are presented in Tables 6 and 7. For the baseline scenario (Table 6), we have the result of the selection of t_1, t_4, t_3, while for scenario (b) we get the selection of t_1, t_4, t_2 (Table 7).

Table 6
Risks and priorities of the baseline scenario (a)

Task ID	1-U	Risks R	Priorities P
t_1	0.10	0.54	11.5
t_2	0.30	1.24	6.8
t_3	0.20	0.60	8.4
t_4	0.05	0.44	10.6
t_5	0.60	1.44	3.6

Table 7
Risks and priorities of scenario (b) with reduced description quality indicators t_2 and t_3

Task ID	New clarity of description U	1-U	Risks R	Priorities P
t_1	0,90	0.10	0.54	11.5
t_2	0,50	0.50	1.9	6.1
t_3	0,60	0.40	1.20	7.8
t_4	0,95	0.05	0.44	10.6
t_5	0,4	0.60	1.44	3.6

The next step was to assign tasks to developers. The assignment rule is shown in Table 8 provided for the sum between the availability of skills (1 if the developer has all the necessary skills; 0.5 - partially; 0 - absent) and priority (1 for the first priority, 0.5 - for the second).

Table 8
Assignment of tasks to performers

Task ID	d_1	d_2	d_3
t_1	1 + 1 = 2.0	0 + 0 = 0	1 + 0.5 = 1.5
t_4	1 + 0.5 = 1.5	0 + 0 = 0	1 + 1 = 2.0
t_3	0 + 0 = 0	1 + 1 = 2.0	0 + 0 = 0
t_2	1 + 0 = 1.0	0 + 0 = 0	1 + 0 = 1.0

In scenario (a): $t_1 \rightarrow d_1$, $t_4 \rightarrow d_3$, $t_3 \rightarrow d_2$, in scenario (b): $t_1 \rightarrow d_1$, $t_4 \rightarrow d_3$, $t_2 \rightarrow d_1$.
A comparison of the results is shown in Table 9.

Table 9
Comparison of results

Indicator	Scenario A	Scenario B
Selected tasks for the sprint	t_1, t_2, t_3	t_1, t_4, t_2
Σ value	25.9	24.2
Σ risk	2.58	3.88
Function W	23.32	20.32
Hours used /48	18/48	14/48
Is the distribution balanced (are all developers involved)	All developers are involved	Developer d2 has no tasks

The results are visually presented in Fig. 4.

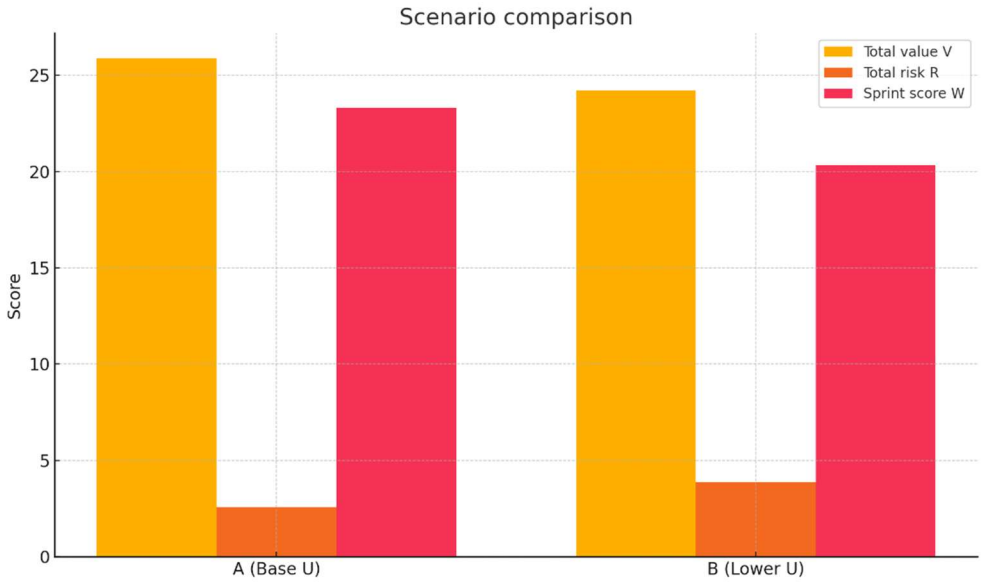

Figure 4: Comparison of sprint metrics for two scenarios.

It can be seen that even with a slight decrease in the clarity of only two tasks (scenario B), the risk increases by ≈ 50%, which reduces 3 points of the integrated benefit.

Thus, we can conclude that the proposed model significantly responds to the quality of task description - its deterioration, even for one task, can exclude it from the sprint and significantly worsen the overall W. The weightsγ_1 andγ_1 allow the manager to strengthen the role of textual clarity. You can also see that only those assignments remain in the plan that simultaneously meet the developers' skill and desire, thus increasing motivation and sustainability of the distribution

Thus, even with a small example, the model clearly shows why some tasks should be postponed and some should be redefined to avoid losing value and creating risks.

5. Conclusion and Future Work

This paper has identified the factors that increase the level of project defects. We systematically analyzed the impact of the main factors (developers' qualifications, inaccurate assessment of sprint tasks, inaccurate task description, reduced team resources, and internal team culture) on increasing

the likelihood of defects and sprint failures. This made it possible to lay the groundwork for their targeted control and elimination.

A framework for intelligent planning of the IT project team's work is proposed, based on the use of evaluation, forecasting, and recommendation systems, which specifies how data flows are transformed into an optimized sprint plan. The framework is based on the integration of artificial intelligence models with Agile management practices, which allows for sound decision-making support, increasing the efficiency of teamwork and the quality of the software product.

The proposed model of intelligent planning of the IT project team formalizes the process of selecting tasks for sprinting, taking into account a set of critical factors that affect the efficiency of team activities. The created model integrates historical metrics, business value of tasks, and current risks, and then, using ML/LLM algorithms, generates recommendations for the optimal content of the sprint. This way, the manager receives an analytically sound plan that minimizes overload and increases the predictability of deadlines. A method for improving textual descriptions of project tasks is proposed, which allows to increase the accuracy of task perception and reduce the risk of their failure by using machine learning models and large-scale language models to evaluate and improve the text. The method uses a BERT-based classifier and LLM reformulation rules to automatically assess the clarity, completeness, and unambiguity of each task. If necessary, the system offers an edited version, increasing the clarity of requirements for developers and thereby reducing the number of defects arising from unclear descriptions.

The research goal was achieved, namely, reducing the level of defects in project performance by using models and a method of intelligent planning of the IT project team's work based on a flexible methodology. Experimental verification of the proposed model and method on control sets confirmed the sensitivity of the model to the quality of the text description of the tasks: reducing the clarity of only two descriptions increases the aggregate risk of defects by 50% and reduces the integral benefit by 10–15%. The applied approach of automatic selection of stable task assignment, on the contrary, preserves the value of the sprint and eliminates blocking pairs without additional time costs. Thus, the proposed approach allows to increase the value of the sprint, managing only the quality of the text and risk weights.

Acknowledgements

The work is funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under project No. 09I03-03-V01-00115.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Agile software development manifesto. URL: https://agilemanifesto.org/iso/uk/manifesto.html.

[2] The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100.

[3] M. Rohovyi, M. Grinchenko, Towards the improvement of project team performance based on large language models. In: Radioelectronic and computer systems, №. 4 (112), 2024, pp.229–247. doi:10.32620/reks.2024.4.19.

[4] M. Rohovyi, M. Grinchenko, Project team management model under risk conditions. In: Bulletin of NTU "KhPI". Series: Strategic management, portfolio, program and project management, 2023, № 1 (7), pp. 3-11. doi: 10.20998/2413-3000.2023.7.1.

[5] M. Rohovyi, M. Grinchenko, Comparative Analysis of stable matching algorithms for intelligent work planning of IT teams. In: Bionics of intelligence, Kharkiv: KNURE, №2 (101), 2024, pp.56–63. doi:10.30837/bi.2024.2(101).09.

[6] J. Menezes, C. Gusmão, H. Moura, Risk factors in software development projects: a systematic literature review. In: Software Qual Journal, 2019, Vol. 27, pp. 1149–1174. doi:10.1007/s11219-018-9427-5.
[7] V. Kolychev, N. Bezmenskii, Estimation of the tasks complexity for large-scale high-tech projects using Agile methodologies. In: Procedia Computer Science, 2018, vol. 145, pp. 266-274. doi:10.1016/j.procs.2018.11.057.
[8] E. Weflen, C. A. MacKenzie, I. V.Rivero, An influence diagram approach to automating lead time estimation in Agile Kanban project management. In:Expert Systems with Applications, 2022, vol. 187, 115866, pp. 1-12. doi:10.1016/j.eswa.2021.115866.
[9] H. Femmer, D. Fernández, S. Wagner, S. Eder, Rapid quality assurance with Requirements Smells. In: Journal of Systems and Software, 2017, vol. 123, pp. 190–213. doi:10.1016/j.jss.2016.02.047.
[10] A. Casamayor, D. Godoy, M. Campo, Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. In: Information and Software Technology, 2010, vol. 52, issue 4, pp. 436–445. doi:10.48550/arXiv.1611.08847.
[11] M. Urbieta etc., The impact of using a domain language for an agile requirements management. In: Information and Software Technology, 2020, vol. 145, pp. 1–16. doi:10.1016/j.infsof.2020.106375.
[12] Cherednichenko, O.; Nebesky, L. and Kováč, M. (2024). Gathering and Matching Data from the Web: The Bibliographic Data Collection Case Study. In Proceedings of the 21st International Conference on Smart Business Technologies, ISBN 978-989-758-710-8, ISSN 2184-772X, pages 139-146. DOI: 10.5220/0012863500003764.
[13] Z. Li, X. Wang, W. Jing, J. Wu, Z. Zhang, etc. A Unified Understanding of Deep NLP Models for Text Classification URL: https://arxiv.org/abs/2206.09355.
[14] R. Ishizuka, H. Washizaki, N. Tsuda, Y. Fukazawa, S. Ouji, S. Saito, Y. Iimura, Categorization and Visualization of Issue Tickets to Support Understanding of Implemented Features in Software Development Projects. In: Applied Sciences, 2022, №. 12, iss. 7, p. 3222. doi:10.3390/app12073222.
[15] P. Chawla, S. Hazarika, H.-W. Shen, Token-wise sentiment decomposition for convnet: Visualizing a sentiment classifier. In: Visual Informatics, 2020, vol. 4, issue 2, pp. 132–141. doi:10.1016/j.visinf.2020.04.006.
[16] F. Dalpiaz, P. Gieske, A. Sturm, On deriving conceptual models from user requirements: An empirical study. In: Information and Software Technology, 2021, vol. 131, 106484, pp. 1–13. doi:10.1016/j.infsof.2020.106484.
[17] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, vol. 1, pp. 4171-4186. doi:10.48550/arXiv.1810.04805.
[18] Kopp, A., Orlovskyi, D., Orekhov, S. An Approach and Software Prototype for Translation of Natural Language Business Rules into Database Structure. CEUR Workshop Proceedings, 2021, 2870, pp. 1274-1291. http://ceur-ws.org/Vol-2870/paper94.pdf.
[19] A. Briciu, G. Czibula, M. Lupea, A study on the relevance of semantic features extracted using BERT-based language models for enhancing the performance of software defect classifiers. In: Procedia Computer Science, 2023, vol. 225, pp. 1601-1610. doi:10.1016/j.procs.2023.10.149.
[20] A. Abdu, Z. Zhai, R. Algabri, H. Abdo etc, Deep learning-based software defect prediction via semantic key features of source code -systematic survey. In: Mathematics, 2022, vol. 10, no. 17, p. 3120. doi:10.3390/math10173120.
[21] H. Moharil, A. Sharma, TABASCO: A transformer based contextualization toolkit. In: Science of Computer Programming, 2023, vol. 230, p. 102994. doi:10.1016/j.scico.2023.102994.
[22] N. N. Tuan, H. Q. Hang, Iteration Scheduling Using Bayesian Networks in Agile Software Development. In: Proc. of Vietnamese Academic Workshop, 2017, pp. 300-308. doi:10.15625/vap.2017.00038.
[23] F. Berhanu, E. Alemneh, Classification and Prioritization of Requirements Smells Using Machine Learning Techniques. In: International Conference on Information and Communication

Technology for Development for Africa (ICT4DA), IEEE, 2023, pp. 49-54. doi:10.1109/ICT4DA59526.2023.10302263.
[24] W. Han, Validating differential relationships between risk categories and project performance as perceived by managers. In: Empir Software Eng, 2014, pp. 1956–1966. doi:10.1007/s10664-013-9270-z.
[25] S. Sundararajan, B. Marath, P. Vijayaraghavan, Variation of risk profile across software life cycle in IS outsourcing. In: Software Qual Journal, 2019, № 27 pp. 1563–1582. doi:10.1007/s11219-019-09451-8.
[26] M. Marinho, S. Sampaio, H. Moura, Managing uncertainty in software projects. In: Innovations Syst Softw Eng., 2018, vol.14, pp. 157–181. doi: 10.1007/s11334-017-0297-y.
[27] M. Uzzafer, A simulation model for strategic management process of software projects. In: Journal of Systems and Software, 2013, vol. 86, iss. 1, pp. 21-37. doi:10.1016/j.jss.2012.06.042.
[28] R. Adel, H. Harb, A. Elshenawy, A Multi-agent Reinforcement Learning Risk Management Model for Distributed Agile Software Projects. In: Tenth International Conference on Intelligent Computing and Information Systems (ICICIS), 2021, pp. 512-520. doi:10.1109/ICICIS52592.2021.9694252.
[29] G.D. Taye, Y.A. Feleke, Prediction of failures in the project management knowledge areas using a machine learning approach for software companies. In: SN Appl., 2022, sci. 4, 165, pp. 2523-3971. doi:10.1016/j.ress.2021.107864.
[30] D. Li, L. Deng, X. Zeng, Z. Cai, Dynamic simulation modelling of software requirements change management system. In: Microprocessors and Microsystems, 2021, vol. 83, pp. 1-6. DOI: 10.1016/j.micpro.2021.104009.