UDC 004.021:005.8 DOi 10.30837/ bi.2024.2(101).09

M. Rohovyi[1], M. Grinchenko[2]

[1]NTU "KhPI", Kharkiv, Ukraine, nikrogovoy@gmail.com,
ORCID ID: 0000-0002-7902-3592

[2]NTU "KhPI", Kharkiv, Ukraine, marina.grynchenko@khpi.edu.ua,
ORCID ID: 0000-0002-8383-2675

COMPARATIVE ANALYSIS OF STABLE MATCHING ALGORITHMS FOR INTELLIGENT WORK PLANNING OF IT TEAMS

The article is devoted to a comparative analysis of stable mapping algorithms for intelligent planning of work of IT teams working according to agile development methodologies. The authors consider the problem of effective task distribution between developers as a problem of finding stable mapping. The paper presents an overview of existing approaches to task distribution in project teams and justifies the relevance of studying stable mapping algorithms in this context. A research methodology is proposed, which includes the creation of a simulation environment for modeling the task distribution process, synthetic data generation, implementation and evaluation of five key algorithms: SOSM, EADAM, MESMA, RSD and TTC. The purpose of the study is to formulate recommendations for the implementation of stable mapping algorithms for planning and task distribution in IT teams using agile project management methodologies. According to the results of the experiments, the EADAM and SOSM algorithms are recommended for practical application due to their balance of stability, efficiency and satisfaction of performers.

TASK DISTRIBUTION, STABLE COMPARISON ALGORITHMS, IT TEAM, FLEXIBLE DEVELOPMENT METHODOLOGIES, PROJECT MANAGEMENT

Роговий М., Гринченко М. Порівняльний аналіз алгоритмів стабільного зіставлення для інтелектуального планування роботи ІТ-команд. Стаття присвячена порівняльному аналізу алгоритмів стабільного зіставлення для інтелектуального планування роботи ІТ-команд, що працюють за гнучкими методологіями розробки. Автори розглядають проблему ефективного розподілу завдань між розробниками як задачу знаходження стабільного зіставлення. У роботі представлено огляд існуючих підходів до розподілу завдань у проєктних командах та обгрунтовано актуальність дослідження алгоритмів стабільного зіставлення в цьому контексті. Запропоновано методологію дослідження, що включає створення симуляційного середовища для моделювання процесу розподілу завдань, генерацію синтетичних даних, реалізацію та оцінку п'яти ключових алгоритмів: SOSM, EADAM, MESMA, RSD та TTC. Метою дослідження є формування рекомендацій щодо імплементації алгоритмів стабільного зіставлення для планування та розподілу задач в ІТ-командах, що використовують гнучкі методології управління проєктами. За результатами проведених експериментів алгоритми EADAM та SOSM рекомендовані для практичного застосування через їх баланс стабільності, ефективності та задоволення виконавців.

РОЗПОДІЛ ЗАВДАНЬ, АЛГОРИТМИ СТАБІЛЬНОГО ЗІСТАВЛЕННЯ, ІТ-КОМАНДА, ГНУЧКІ МЕТОДОЛОГІЇ РОЗРОБКИ, УПРАВЛІННЯ ПРОЄКТАМИ

Introduction

One of the most common approaches to managing software development teams is the agile Scrum methodology. The team's work is usually organized in iterations with weekly sprints. The team breaks down each task into separate tasks, for which they estimate the time for completion. When assigning tasks to developers, it's important to consider their skills and preferences. Effectively assigning developers to tasks is a critical challenge in software development teams. The goal is to assign tasks to developers in a way that maximizes productivity, ensures job satisfaction, and maintains team stability.

The Scrum methodology regulates the distribution of tasks based on the results of team discussions, considering the interests of the performers, the wishes of the manager, and the priorities of the task for the project. As a result, there is a risk of failure to complete or incorrectly complete a task due to the choice of an inappropriate performer. In addition, if the task was assigned by the project manager, but there are conflicts with the interests of the performer, there is a risk of failure to complete the task due to the emergence of another, higher priority from the performer's point of view. This problem is similar to the well-known stable matching problem, which is to find a match between two sets of elements (e.g., developers and tasks) such that there are no two elements that prefer each other over their current choice.

So, in a general sense, the task allocation problem can be viewed as the problem of forming stable pairs between executors and sprint tasks. Previous research has studied various aspects of task assignment and developer performance. However, there is a gap in the application of stable matching algorithms specifically tailored to the context of IT teams, considering factors such as developer skills, task complexity, and developer interests, which is important in the context of agile project management methodologies. This paper aims to fill this gap by conducting a comparative analysis of several stable matching algorithms, evaluating their effectiveness in different team scenarios.

1. Literature review

The task allocation is an important step in project management, especially in agile methodologies. Effective task distribution helps to optimize teamwork, increase productivity, and ensure that tasks are completed on time. Let's look at the existing approaches to the distribution of tasks among project executors in the works of various researchers.

The authors of the paper [1] propose an algorithm for assigning employees to project work under conditions of uncertainty, which considers the level of professional competence of the staff and the qualification requirements for project tasks. They define the main parameters of staff assessment, which include not only the available knowledge and experience but also the personal qualities of the employee. The researchers emphasize the importance of correlating the requirements for project tasks and the qualifications of labor resources, which can contribute to a more efficient distribution of tasks in the context of a flexible IT project development methodology.

Paper [2] presents a decision support model using a genetic algorithm for task allocation. The main entities used are tasks, resources, goals, and various parameters. Based on the genetic algorithm, a chromosome is formed for double fragments, first information on the allocation of resources to individual tasks and then information on the allocation of time scales to a separate combination of task resources.

In the work [3], researchers presented an approach to supporting the distribution of tasks in distributed teams using multicriteria decision analysis (MCDA). The study is based on a real-world example where a multicriteria model was created to support the distribution of work in distributed teams. This work offers a structured approach to solving the complex problem of task assignment in globally distributed software development projects.

The authors of the paper [4] presented a method for assigning tasks for crowdsourcing software in the context of collaborative development. The authors proposed an approach to task assignment in a crowdsourcing environment, which is a promising model of software development. This work is aimed at solving the problem of efficient task assignment in crowdsourced software development projects, which is an important aspect for the successful implementation of such projects. The results show that the proposed method can increase the utility by about 25% and the average success rate by about 30% compared to the sequential assignment method.

Three algorithms were proposed in [5] to solve the problem of task distribution: GAN (Generative Adversarial Networks) for text generation, decision-making data generation, and data function enhancement; Baum-Welch algorithm for obtaining model parameters; and Viterbi algorithm for obtaining an optimal task assignment strategy. Based on these algorithms, efficient task allocation strategies are created to maximize the total value of tasks performed by employees.

An integrated artificial intelligence system [6] provides dynamic data-driven optimization of resource and task allocation to improve the productivity of software projects. This system includes natural language processing (NLP), reinforcement learning, and Bayesian networks. It generates task requirements from project documents, predicts the optimal resource allocation using reinforcement learning, and validates the allocation using a Bayesian network trained on past project data.

In [7], the authors propose an extended method and algorithm by combining optimized flexible iteration scheduling and the ability to predict and manage risks in resource-limited Bayesian networks. Based on the method, software is developed as an auxiliary tool for managers to control their project schedules. The tool also provides a robust set of strategies for sequencing the task of flexible iteration scheduling.

In the paper [8], the authors reviewed various approaches to task allocation adopted by agile software development researchers in a quantitative manner. The table shows a comparison of accepted task allocation applications along with their pros and cons. The study found that most approaches to task allocation are quantitative, but the qualitative aspect has not been considered to a large extent during this process.

In order to develop a flexible and efficient model for planning a software project, an event-based scheduling (EBS) approach and ant colony optimization (ACO) algorithm were developed in [9]. The proposed method allows modeling resource conflicts and maintains flexibility in the allocation of human resources. The results of 83 experiments demonstrate the prospects of the proposed method.

A multi-criteria decision-making model for planning and fine-tuning project plans [10] was developed using cognitive mapping and MACBETH (measuring attractiveness using category-based evaluation technique). The proposed model was based on the value judgments of decision makers, which makes the model subjective.

In [11], the authors combined several factors used in task assignment and determined their importance, allowing them to establish a priority order among them. The paper presents a hybrid methodology based on VDA techniques to classify and organize the factors that guide the assignment of tasks to distributed teams in software development projects. Tasks were grouped according to their type, requirements, architecture, implementation, and testing. This method involves classifying and organizing the factors that govern task assignment in a distributed scenario.

The authors of [12] proposed a dynamic task assignment algorithm (DUTA) and a dynamic crowdsourcing software algorithm based on utilities. The Kuhn-Munkress

method and the weighted bipartite graph algorithm are used to determine the optimal match between tasks and workers. Experimental results showed that DUTA gave satisfactory performance results for the overall allocation utility with a better task allocation success rate than the user reliability-based algorithm. DUTA achieved an average allocation accuracy of 85.63%, which demonstrates effective task management.

In [13], the authors propose a sprint planning decision support system (SPESS), which is a tool to help managers plan sprints. SPESS uses poker for planning and the Hungarian algorithm as a basis, and in addition to considering the experience factor, it considers the level of developer competence and task dependencies. The result is comprehensive and accurate sprint planning for fast and high-quality product delivery.

Paper [14] proposes an approach to task assignment in a Scrum team using multi-agent modeling based on p-values. The author has developed a task assignment algorithm that uses p-values as a key factor in making decisions about assigning tasks to agents. The p-value is seen as a relative view of the agent and the task it is working on. This approach allows you to effectively distribute tasks in a Scrum team, considering the characteristics of agents and the characteristics of tasks.

Researchers in [15] use mixed integer nonlinear programming (MINLP) to plan a project and solve the problem of staff allocation with a time-dependent learning effect based on task similarity. The learning effect of a task depends on the time when project staff start performing that task. If project staff performs repetitive and/or similar tasks, then these staff can gain experience and complete tasks faster than planned. Thus, the sequence of tasks is important to assign to project staff to minimize project completion time while considering the similarity of tasks in terms of learning.

The work [16] is aimed at building a goal tree, which allowed to reflect the overall goal and subgoals that must be ensured for the efficient allocation of resources. As a mathematical model, a Boolean integer programming problem was used, which with a sufficient degree of adequacy was able to reflect the realities of project portfolio formation in conjunction with the allocation of resources between the relevant projects in the portfolio. The result of the work was a prototype software that implements resource allocation modeling using the Balash method.

According to the results of the research published in [17], a new approach to the development of an integrated resource and task allocation optimization system (RATAOS) using enterprise architecture is proposed to improve the efficiency of project management in IT companies. The main aspects of their research include integration of a project management information system (PMIS) with an optimization system developed using a random forest model and natural language processing (NLP). Optimization of resource and task allocation resulted in a 14% reduction in operating costs and 88.7% reduction in planning phases. The effectiveness of the proposed system is demonstrated by a 50.80% reduction in project completion time [17].

The authors of [18] propose a comprehensive methodology for forming an IT project team based on solving a multi-criteria distribution problem using metrics, scheduling, and calculating employee workload. Two team formation templates are proposed: for a project and for a task within a project. Such an integrated approach reduces the time for forming a project team and eliminates the risk of misassignment.

Thus, current research in the field of task allocation in a project team shows a tendency to integrate various approaches, including multi-criteria models, algorithmic solutions, and artificial intelligence methods. These approaches are aimed at increasing the efficiency of project management, optimizing the use of resources, and improving the quality of project results.

The literature review reveals various approaches to task allocation in project teams. However, not enough attention has been paid to the study of task allocation in a project team based on the stable comparison problem. In our opinion, such a study is important in terms of ensuring a stable distribution, which is especially relevant for a scrum team, balancing the distributed tasks between performers, and optimizing resources, considering the interests and competencies of team members.

2. Purpose and objectives of the study

The purpose of the study is to formulate recommendations for the implementation of stable matching algorithms for planning and distributing tasks by the project team when using flexible project management methodologies.

To achieve this goal, it is proposed to select stable matching algorithms that have successfully proven themselves for solving problems in other subject areas; define criteria for evaluating the effectiveness of algorithms; conduct experiments on model data and analyze the prospects for implementing this task distribution formulation in a project team that uses flexible management methodologies.

3. Research methodology

This section presents a methodology developed for the comparative analysis of task allocation algorithms in IT teams working with agile development methodologies. Our study aims to evaluate the effectiveness of different stable matching algorithms in the context of IT project team scheduling.

The basis of the study is the creation of a simulation environment that allows modeling the process of distributing tasks among developers, considering realistic conditions and constraints typical of IT projects. We consider five key algorithms: SOSM, EADAM, MESMA, RSD,

and TTC. The proposed approach includes generating synthetic data that simulates developer and task profiles, calculating compatibility metrics, building preference lists, implementing algorithms, and evaluating them using a number of metrics. This allows us to conduct a comprehensive analysis of the effectiveness of each algorithm in different scenarios and conditions. This approach provides a solid basis for comparing algorithms and formulating recommendations for their application in real-world IT projects.

3.1 Overview of algorithms

The Student-Optimal Stable Mechanism (SOSM) algorithm is based on the Gale-Shapley deferred acceptance algorithm [19]. In this context, developers are considered as students who propose to tasks (schools) to form a pair in the order of their preference lists. Each task also has a list of preferred developers. The algorithm runs in several rounds.

1. Proposal phase: developers rank the tasks according to their preferences and each developer "offers" himself or herself to the most preferred task.

2. Acceptance phase: Tasks have a prioritized list of developers. Each task reviews the proposals and temporarily accepts the highest priority developer and rejects the others.

3. Iteration: Rejected developers move on to the next task on their list.

The process continues until all developers have been assigned or rejected by all tasks. SOSM guarantees a stable matching that is optimal for developers, meaning that no developer can get a better assignment without making someone else worse off. This promotes fair distribution and increases developer motivation.

The Efficiency-Adjusted Deferred Acceptance Mechanism (EADAM) algorithm [20, 21], compared to SOSM, strives to achieve Pareto-efficiency, but may violate stability to improve efficiency. EADAM can lead to better overall satisfaction of developers with their assignments compared to SOSM. In the context of assigning tasks to developers in IT projects, the EADAM algorithm can be described as follows:

1. First, the standard deferred acceptance algorithm is applied to obtain the initial allocation.

2. Interrupters are identified and eliminated iteratively - these are pairs (developer, task) where a developer offers himself to a task, causes another developer to be rejected, but later gets rejected himself. As a result, no developer can get a better assignment without making the situation worse for the others.

3. Repeat the process until all interrupters are removed.

EADAM strives to find a balance between efficiency and fairness in task allocation, which can be useful in a dynamic IT project environment.

The Maximally Efficient and Stable Matching Algorithm (MESMA) [22] aims to find a stable distribution of tasks among developers with the maximum overall weight (efficiency) and focuses on maximizing the overall matching efficiency while ensuring stability. The inputs are a system of developer preferences for tasks (and vice versa) and a weighting function that determines the efficiency of each possible assignment. MESMA uses a linear programming approach to solve the problem of maximizing the weight of a stable match. However, the algorithm can be computationally challenging for large projects, as the maximum weighted stable matching problem is NP-hard.

Random Serial Dictatorship (RSD) [23, 24] is a simple, strategically secure algorithm that assigns developers to tasks based on a randomly determined order. The first developer in the sequence chooses the task that is of the highest priority for him or her from the entire set of available tasks. The second developer chooses his or her highest priority task from the remaining ones. The process continues until all developers have selected a task or until the available tasks run out.

The advantage of RSD is strategic security, i.e., developers have no incentive to misrepresent their preferences. RSD provides a simple and fair method of task allocation but may require additional mechanisms to optimize the efficiency of allocation in the context of IT projects.

The Top Trading Cycles (TTC) algorithm [25] also allows optimizing the distribution of tasks based on the wishes of developers, provides Pareto-efficient distribution, and is strategically safe. The algorithm works iteratively: each developer indicates the most preferred task from the list of available ones. Cycles are formed where developers point to each other through their preferred tasks. For each identified cycle, tasks are exchanged between developers. Developers who participated in the exchange are removed from further consideration along with their new tasks. The process is repeated for the remaining developers until all developers have been assigned or there are no more opportunities for exchange.

Thus, five algorithms were selected for the comparative analysis, which allow optimizing the distribution of tasks based on the wishes of developers, their qualifications, and preferences from the project's point of view.

3.2 Framework for modeling task distribution

To conduct experiments with the five matching algorithms (SOSM, EADAM, MESMA, RSD, TTC), we developed a comprehensive simulation framework that models the process of task allocation in IT teams. Our framework consists of the following key components:

– A synthetic data generator that creates realistic developer profiles with attributes such as skills, experience, workload, and task characteristics (such as complexity, priority, etc.).

– A compatibility calculation module that calculates the compatibility score between each developer-task pair.

– Generator of preference lists for developers and tasks.

– A module for implementing matching algorithms that implements the five algorithms considered (SOSM, EADAM, MESMA, RSD, TTC).

– Algorithm performance evaluation module. We have chosen the following metrics to evaluate algorithms:

1. Total Compatibility Score (TCS), which is calculated as the sum of all compatibility values between assigned developers and tasks:

$$S_{total} = \sum_{(d,t)\in M} C(d,t),$$

where M is a set of pairs (developer, task) in a matching, $C(d, t)$ is the compatibility score between the developer d and the task t.

2. Number of blocking pairs (Blocking Pairs). The pair (d,t) is considered to be a blocking pair if the developer d prefers the task t over his current assignment and the task t prefers the developer d over its current assignment. The number of blocking pairs is calculated as follows:

$$B = \sum_{(d,t)\notin M} I\left[(t \succ_d M(d)) \wedge (d \succ_t M(t))\right],$$

where $I[\cdot]$ is an indicator function equal to 1 if the condition is met and 0 otherwise, $M(d)$ is the task assigned to the developer d, $M(t)$ is the developer assigned to the task t. $t \succ_d M(d)$ means that the task t is more desirable for the developer d than the one assigned to him, $d \succ_t M(t)$ means that the developer d is more desirable for the task t than the one it was assigned.

3. Developer Satisfaction is defined as the average rank of assigned tasks in the developer preference lists using the formula:

$$S_{dev} = \frac{1}{|D|}\sum_{d\in D} rank_d\left(M(d)\right),$$

where D is the set of all developers, $rank\_d\ (M(d))$ is the position of the task t in the preference list of the developer d, $M(d)$ is the task assigned to the developer d.

The lower S_{dev} value means higher developer satisfaction.

4. Execution time — the computational time required by each algorithm to reach a solution.

The proposed framework allows conducting large-scale experiments with different team and project configurations, providing an in-depth analysis of the effectiveness of each algorithm in the context of task allocation in IT projects.

4. Experimental research

We designed experiments to test the algorithms under different conditions, focusing on the following variables:

– Team size: number of developers (small, medium, large).

– Number of tasks varies with the size of the team to simulate different workloads (light, moderate, heavy).

– Skill distribution: Developers have similar skill levels (homogeneous) or developers have diverse skills (heterogeneous)

– Preference structure: correlated (preferences are aligned with skills) or random (preferences are assigned randomly).

We identified eight basic scenarios based on different conditions in order to evaluate the scalability and performance of the algorithms under different workloads. To test the impact of agreed or random preferences on the algorithms' results, we added the corresponding scenarios.

The scenarios are presented in tab. 1. It should be noted that Scenario 5 focuses on the impact of task urgency.

Thus, we have formed the scenarios of experiments for testing the selected algorithms using the developed framework.

Table 1

Description of the experimental conditions

Experiment Number	Name	Team Size	Number of Tasks	Skill Distribution	Preference Structure
1	Exp1_Small_Homogeneous_Light	5	5	homogeneous	correlated
2	Exp2_Small_Heterogeneous_Light	5	5	random	correlated
3	Exp3_Medium_Heterogeneous_Moderate	15	30	random	correlated
4	Exp4_Medium_Homogeneous_Random	15	15	homogeneous	random
5	Exp5_Sensitivity_Urgency	15	30	random	correlated
6	Exp6_Large_Heterogeneous_Heavy	30	90	random	correlated
7	Exp7_Large_Homogeneous_Random	30	30	homogeneous	random
8	Exp8_Large_Heterogeneous_Moderate	30	30	random	correlated

5. Results and analysis

The results of the experiments conducted under the defined scenarios are shown in tab. 2. Let's take a closer look at the results for different algorithms and scenarios. In terms of the overall compatibility score, the EADAM and SOSM algorithms consistently achieved high overall compatibility scores in all experiments, indicating effective comparisons. It is worth noting that MESMA slightly outperformed EADAM and SOSM in terms of compatibility, which is explained by the optimization orientation of the algorithm. Therefore, we can conclude that EADAM and SOSM are effective in creating comparisons, while MESMA provides minor improvements due to increased computational complexity.

In terms of stability (Blocking Pairs), EADAM and SOSM provided stable matches with zero blocking pairs in all scenarios, RSD did not guarantee stability, and TTC resulted in a large number of blocking pairs, especially in larger and more complex scenarios. It should be noted that the MESMA algorithm did not explicitly report blocking pairs, so it is difficult to compare it with other algorithms by this indicator.

Table 2

Experimental results

Experiment Number	Algorithm	Total Compatibility Score	Blocking Pairs	Developer Satisfaction	Runtime (s)
1	EADAM	4,30	0	3,00	0,0000
	MESMA	4,30	0	3,00	0,0090
	RSD	4,30	0	3,00	0,0000
	SOSM	4,30	0	3,00	0,0000
	TTC	4,30	4	3,00	0,0000
2	EADAM	3,22	0	1,74	0,0000
	MESMA	3,42	0	1,85	0,0055
	RSD	3,21	0	1,82	0,0000
	SOSM	3,22	0	1,74	0,0000
	TTC	2,58	2	2,17	0,0000
3	EADAM	14,43	0	4,34	0,0001
	MESMA	14,53	0	4,57	0,4510
	RSD	14,45	0	4,35	0,0000
	SOSM	14,43	0	4,34	0,0001
	TTC	8,13	17	10,29	0,0003
4	EADAM	13,00	0	7,42	0,0002
	MESMA	13,00	0	7,42	0,3780
	RSD	13,00	0	7,47	0,0000
	SOSM	13,00	0	7,42	0,0001
	TTC	13,00	52	7,99	0,0003
5	EADAM	14,43	0	4,34	0,0001
	MESMA	14,53	0	4,57	0,4435
	RSD	14,45	0	4,35	0,0000
	SOSM	14,43	0	4,34	0,0001
	TTC	8,13	17	10,29	0,0003
6	EADAM	30,76	0	8,34	0,0004
	MESMA	30,84	0	9,00	19,9934
	RSD	30,77	0	8,27	0,0000
	SOSM	30,76	0	8,34	0,0002
	TTC	18,53	87	27,37	0,0022
7	EADAM	26,04	0	14,16	0,0008
	MESMA	26,04	0	14,16	6,7477
	RSD	26,04	0	14,13	0,0001
	SOSM	26,04	0	14,16	0,0004
	TTC	26,04	222	15,57	0,0014
8	EADAM	23,59	0	7,76	0,0003
	MESMA	24,49	0	8,38	1,9629
	RSD	23,64	0	7,76	0,0000
	SOSM	23,58	0	7,76	0,0002
	TTC	18,81	92	10,25	0,0007

It can be concluded that stability is a significant issue with TTC, making EADAM and SOSM preferable when stability is critical. From the perspective of Developer Satisfaction, algorithms that ensure stability also contribute to higher developer satisfaction.

The execution time of the algorithms is shown in Fig. 1, where we can clearly see that MESMA has a significantly higher execution time, especially in experiments with large teams (almost 20 seconds in Experiment 6), which is explained by its computational complexity.

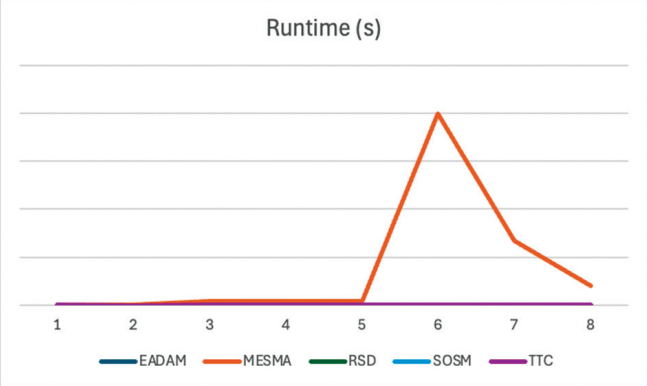

Fig. 1. Time of execution

Regarding the other algorithms, we can note that EADAM and SOSM were highly efficient, with execution times ranging from microseconds to milliseconds (tab. 2), RSD was the fastest algorithm due to its simplicity, and TTC had an average execution time that increased with the size of the problem. Therefore, while MESMA may offer minor performance improvements, its computational cost may not justify its use in time-sensitive environments.

The results are summarized in Fig. 2. The EADAM and SOSM algorithms provided stable and efficient matching. At the same time, TTC instability became clearer with increasing team size, with a significant increase in blocking pairs and lower compatibility scores, and MESMA's runtime increased, raising scalability concerns.

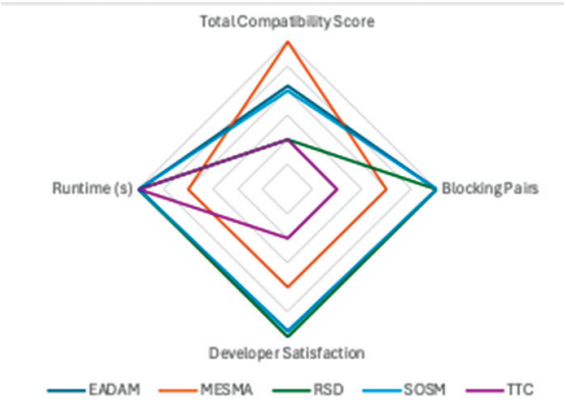

Fig. 2. Comparative analysis

In the fourth and seventh scenarios (homogeneous skills, random preferences), all algorithms achieved similar compatibility scores. TTC showed limitations in complex scenarios, even with homogeneous skills, the preference structure affects its stability.

Thus, based on the results of the experiments, EADAM and SOSM are recommended for practical use because of their balance of stability, efficiency, and satisfaction. Despite its minor advantages, MESMA has certain computational requirements that limit its practical application. TTC's instability and lower satisfaction levels make it less suitable, especially in heterogeneous and larger teams. The lack of stability of RSD is a critical drawback despite its simplicity and speed.

6. Discussion and conclusions

Agile software development requires effective organizational decisions during the execution of project tasks. Successful task assignment is a challenging management problem in agile software development.

Task assignment decisions are critical to the success of agile teams, but they are not well understood. Traditional survey-based methods limit the scope and level of detail of data collection. Quality, productivity, and motivation are negatively impacted by a lack of transparency and lack of justification for the form of task assignment.

An analysis of different approaches to task distribution allows us to identify current trends in task distribution among project executors, overcome uncertainties, and improve overall project efficiency. In studies [1, 13], the distribution is based on competencies. The authors consider solving this problem by aligning project tasks with the qualifications and competencies of team members. In [2, 9], genetic algorithms are also used to distribute tasks among project executors. These approaches effectively resolve resource conflicts and optimize planning by modeling different scenarios and finding optimal solutions. Studies [4, 12] utilize the potential of crowdsourcing and distributed teams. The proposed models consider employee activity, task complexity, and dependencies between modules to increase the efficiency of cooperation. These approaches demonstrate significant improvements in resource utilization and task completion speed.

Papers [5, 6, 7, 17] use artificial intelligence and machine learning methods. The use of multi-criteria models [3, 10, 18] provides a structured approach to task allocation. By evaluating several factors simultaneously, these models ensure balanced decision-making and integrate qualitative and quantitative metrics. Proposed integrated systems for task optimization [15, 18] emphasize the importance of combining task distribution with resource optimization. Such systems reduce operating costs and shorten planning time, accelerating the delivery of project results.

The reviewed studies emphasize the multifaceted nature of task distribution in Agile project management. The comparative study emphasizes the importance of choosing appropriate stable matching algorithms for task assignment in IT teams. The EADAM and SOSM algorithms

prove to be the most effective, consistently providing stable and efficient mappings with high developer satisfaction and low computational costs.

Possible directions for further research could be to integrate with real data and validate the results using actual team and task data. In the direction of improving the algorithms, multitasking assignments and dynamic team environments should be considered. Consideration of additional factors such as fairness, workload balance, and long-term impact on team performance also need to be explored in more depth.

List of references:

[1] Myroslava Gladka, Olga Kravchenko, Yaroslav Hladkyi, Sholpan Borashova. Qualification and appointment of staff for project work in implementing IT systems under conditions of uncertainty // Proc. of 2021 IEEE International Conference on Smart Information Systems and Technologies, Astana IT University, Nur-Sultan, Kazakhstan. – 2021. – P. 28-30.

[2] Fernandez J., Basavaraju M. Task allocation model in globally distributed software projects using genetic algorithms // Proc. of the 2012 IEEE Seventh International Conference on Global Software Engineering. – 2012. – P. 181.

[3] Barcus A., Montibeller G. Supporting the allocation of software development work in distributed teams with multicriteria decision analysis // Omega. – 2008. – V. 36. – № 3. – P. 464-475.

[4] Yu D., Zhou Z., Wang Y. Crowdsourcing software task assignment method for collaborative development // IEEE Access. – 2019. – V. 7. – P. 35743-35754.

[5] Yin X., Huang J., He W., Guo W., Yu H., Cui L. Group task allocation approach for heterogeneous software crowdsourcing tasks // Peer-to-Peer Networking and Applications. – 2020. 14. – № 3. – P. 1736-1747.

[6] Nakra V. Enhancing Software Project Management and Task Allocation with AI and Machine Learning // International Journal on Recent and Innovative Trends in Computing and Communication. – 2023. 11. – № 11. – P. 1171-1178.

[7] Tuan N. N., Hang H. Q. Iteration Scheduling Using Bayesian Networks in Agile Software Development // Proc. of Vietnamese Academic Workshop. – 2019. – P. 300-308.

[8] Singh M., Chauhan N., Popli R. A Review on Quantitative Task Allocation in Agile Software Development // Proc. of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur. – 2019. – P. 268-273.

[9] Chen W. N., Zhang J., Member S. Ant colony optimization for software project scheduling and staffing with an event-based scheduler // IEEE Transactions on Software Engineering. – 2013. 39. – № 1. – P. 1-17.

[10] Almeida L. H., Albuquerque A. B., Pinheiro P. R. A multicriteria model for planning and fine-tuning distributed Scrum projects // Proc. of 2011 IEEE Sixth International Conference on Global Software Engineering. – 2011. – P. 75-83.

[11] Simão Filho M., Pinheiro P. R., Albuquerque A. B. Analysis of task allocation in distributed software development through a hybrid methodology of verbal decision analysis // Journal of Software: Evolution and Process. – 2017. – V. 29. – № 7. – P. 1-18

[12] Yu D., Wang Y., Zhou Z. Software crowdsourcing task allocation algorithm based on dynamic utility // IEEE Access. – 2019. – V. 7. – P. 33094-33106.

[13] Alhazmi A., Huang S. A decision support system for sprint planning in scrum practice // Proc. of SoutheastCon. – 2018. – P. 1-9.

[14] Wang Z. P-value based task allocation in a scrum team: a multi-agent simulation // Proc. of the IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). – 2019. – P. 1-4.

[15] Arık O. A. Project scheduling and staff allocation problem with time-dependent learning effect: a mixed integer nonlinear programming approach // A – Applied Sciences and Engineering. – 2019.

[16] Vasyluk A., Basyuk T. Algebras of algorithms for modeling the distribution of resources in IT projects // SISN. – 2023. – V. 13. – P. 156-166.

[17] Pratama I. N., Dachyar M., Pratama N. R. Optimization of Resource Allocation and Task Allocation with Project Management Information Systems in Information Technology Companies // TEM Journal. – 2023. 12. – № 3. – P. 1814-1824.

[18] Nechvoloda L. V., Shevchenko N. Yu. Increasing the efficiency of IT project management with the application of complex methodology distribution of performers for work // Taurida Scientific Herald. Series: Technical Sciences. – 2023. – No. 2. – P. 98-105.

[19] Diebold F., Aziz H., Bichler M., et al. Course Allocation via Stable Matching // Bus Inf Syst Eng. – 2014. – V. 6. – P. 97-110.

[20] Jiao Z., Shen Z. School choice with priority-based affirmative action: A responsive solution // Journal of Mathematical Economics. – 2021. – V. 92. – P. 1-9.

[21] Jiao Z., Tian G. Two further impossibility results on responsive affirmative action in school choice // Economics Letters. – 2018. – V. 166. – P. 60-62.

[22] Diebold F., Bichler M. Matching with indifferences: A comparison of algorithms in the context of course allocation // European Journal of Operational Research. – 2017. – V. 260. – № 1. – P. 268-282.

[23] Brandt F., Greger M., Romen R. Towards a Characterization of Random Serial Dictatorship // arXiv. – 2023.

[24] Aziz H., Brandt F., Brill M. The computational complexity of random serial dictatorship // Economics Letters. – 2013. 121. – № 3. – P. 341-345.

[25] Hong M., Park J. Core and top trading cycles in a market with indivisible goods and externalities // Journal of Mathematical Economics. – 2022. – V. 100. – P. 102627.

The article was delivered to editorial stuff on the 18.12.2024