



Models and Methodologies for Optimizing Software Project Performance: An Integrative Review of Artificial Intelligence-Assisted Planning, Resource Allocation, Human Factors, and Data-Driven Decision Support

Bogdan Rânea^{a*}, Alexandru Marin^a

^a National University of Science and Technology POLITEHNICA Bucharest, Splaiul Independentei 313, Bucharest 061126, Romania

ABSTRACT

Software project performance is influenced by planning quality, resource allocation, human factors, risk, monitoring, and the growing use of artificial intelligence. This paper presents an integrative review of models and methodologies that can support the optimization of software project performance. Literature from project management, software engineering, artificial intelligence, optimization, and human-centered systems was analyzed, with emphasis on studies published between 2020 and September 2026. The review groups existing approaches into planning and scheduling, human-resource allocation, predictive and risk analysis, monitoring and early detection, and adaptive decision support. The synthesis shows that artificial intelligence and optimization methods can improve estimation, task allocation, monitoring, and managerial decision support, but their effectiveness depends on data quality, transparency, organizational context, and human oversight. Evidence from software-intensive engineering applications also shows the value of transferable methods for projects with interacting technical and human constraints. The review concludes that software project performance is best addressed through a continuous feedback process connecting project inputs, intelligent analysis, managerial decisions, and execution data. This integrated perspective can support more adaptive and data-informed software project management.

Keywords: Software project management; Artificial intelligence; Project performance; Resource allocation; Human factors; Decision support.

1. Introduction

Due to factors such as shorter delivery cycles, changing needs, dispersed teams, few resources, and rising technological complexity, software project management has become more challenging. There is a need to update our understanding of project success metrics beyond the traditional time, money, and scope metrics. The quality of decisions, as well as productivity, risk, resource utilization, flexibility, and overall project performance, are being more and more recognized as interrelated aspects in recent studies [1]. Additionally, data-driven methods demonstrate that, when operational data is translated into clear and useful indications, predictive models may bolster managerial decision-making [2].

More and more, planning, forecasting, allocating resources, monitoring, and risk management are being aided by artificial intelligence. While recent evaluations have shown that AI (Artificial Intelligence) may enhance many project management tasks, the actual use of this technology is still contingent upon factors such as human acceptability, organizational preparedness, and data quality [3]. Users' openness to automated analysis is affected by factors such as trust, transparency, and governance in data-intensive decision-support systems, which raises similar problems [4].

Project performance is also being redefined by AI. Although a project may complete on schedule, its performance may be subpar due to factors such as the amount of work it generates, the quality of the software it produces, or the frequency with which it has to be corrected. New research on AI and successful projects reveals that cost and time are still important performance metrics, but they need to be evaluated with broader operational and organizational considerations [5]. Instead of seeing data analysis as a standalone technical activity, software programs that aim to provide tailored and adaptive assistance show how valuable it is to turn user and process data into practical actions [6].

In the context of Agile software development, these concerns take on added significance. Although initial estimating, priority, and resource choices are still needed, agile methodologies make long-term planning less rigid and teams may respond quicker to changing needs. Rather of applying planning via a predefined approach, a recent comprehensive analysis of the front-end phase of Agile projects demonstrated that planning should be customized to the project environment [7]. Concurrently, development initiatives including software architecture and user interaction are becoming more common in intelligent software environments, which include local AI, interactive interfaces, and real-time processing [8].

The development and operations movement, or DevOps, takes this issue all the way through the software development life cycle. Project success is dependent on the integration of technical and organizational skills, as well as the growing interconnectedness of development, testing, deployment, monitoring, and operation [9]. Frameworks for software development that integrate data processing, machine learning, evaluation, execution control, and visualization are one example of this kind of integration [10]. Software modules, data, computational resources, validation activities, and users all have dependencies that project managers must organize.

One of the biggest issues with software project management is estimation. Timelines, personnel, finances, and delivery promises are all affected by inaccurate effort estimates. Although the effectiveness of machine-learning approaches for software work estimate is project-specific, validated, and dependent on dataset quality, feature selection, and validation, their usage is on the rise [11]. As shown by Python-based robotic path-planning systems coupled with simulation platforms [12], similar management issues arise in software-intensive engineering applications where algorithms, simulation environments, validation phases, and physical limitations all need to be coordinated throughout the planning process.

Another aspect that is closely related is risk. Problems that arise during software development or deployment may stem from a variety of causes, including requirements, technology, resources, timetable pressure, organizational circumstances, and failures. Study after study has indicated that software risk management is often handled in silos rather than as a life cycle-spanning integrated approach [13]. The variety of potential threats is further increased by the growing interconnection of software and physical systems. Case in point: home automation apps that are BCI (Brain-Computer Interface) based integrate software processing, embedded devices, communication, user interaction, and real-time control into a single functional system [14].

People also have a significant impact on how well a project turns out. As a knowledge-intensive process, software development is very susceptible to the effects of factors such as workload, communication, emotions, motivation, and team dynamics on both productivity and software quality. Project activities, decision making, cooperation, and development results are all impacted by emotional moods, according to research on Agile software developers [15]. Another context where this link is evident is in human-in-the-loop applications, where the accuracy of algorithms is only one factor among several that determine how well a system performs [16].

A growing body of literature has been focused on software engineers' happiness and the elements inside their organizations that could affect it. Individual, group, and company factors all have a role on software engineers' sense of well-being, which in turn influences their efficiency and effectiveness on the job [17]. Automated analysis may be crafted to bolster operational and safety choices, with the human operator retaining ultimate interpretation, as shown by human-centered monitoring technologies [18].

Another vital link between intelligent decision support and conventional project management is resource allocation and scheduling. When activities in a project compete for limited resources while still adhering to precedence relations, one of the primary formal models is the Resource-Constrained Project Scheduling Problem [19]. System performance may be heavily influenced by the way trajectory continuity and technological restrictions are described, according to software-intensive engineering research [20].

By looking at options that are hard to assess manually, modern optimization tools may make this process better. To reduce project length while maintaining resource limitations, many strategies have been used, including exact algorithms, mathematical programming, heuristics, and hybrid approaches [21]. Further evidence that optimization may be seamlessly integrated into software-intensive engineering processes can be seen in the optimization of complicated robotic trajectories, which have been shown to enhance operational efficiency by reducing unnecessary motion [22].

Better project management is not, however, a byproduct of AI's entrance. A number of obstacles, including those related to technology, organizations, and the environment, have been identified in recent studies on artificial intelligence (AI) in software project planning [23]. In multimodal software systems, such as those involving vision, machine learning, embedded processing, and human instructions, the need for meticulous integration becomes apparent in real time [24]. These cases provide credence to the idea that software project success is contingent upon the integration of data, people, technology, and resources.

Estimation, prediction, project risk, and other decision-support uses of machine learning in software project management have been previously reviewed [25]. Project planning, artificial intelligence, human aspects, optimizing resources, Agile methods, and software-intensive system development all have a lack of cohesive literature. Decisions made in software interact with hardware, controls, simulations, validation, and operational limitations; this integration is evident in complex engineering software [26]. Recent Agile development assessments have shown that rather than being separate management issues, delivery pressure, customer engagement, team conditions, productivity, and software quality are tightly interrelated [27].

Consequently, a unified and succinct picture of the concepts and approaches that boost software project efficiency is required. All of the following aspects of project management—scheduling and planning, allocation of resources, risk, human factors, and data-driven decision support—are considered in this analysis as interrelated parts of the same management challenge. Instead of trying to pin down a single, best technique, we'll demonstrate how many methods may be tailored to specific project requirements. In addition to reviewing the relevant literature, the research arranges these methods into an integrated viewpoint that connects project inputs and limitations, intelligent analysis and optimization, human management judgments, and quantifiable project end results.

2. Review Methodology

Due to the interconnected nature of software project performance, optimization, human aspects, software engineering, project management, artificial intelligence, and decision assistance, this research used an integrated literature review strategy. A more comprehensive understanding of a complicated

research subject may be fostered by the integration of empirical, methodological, and conceptual studies in an integrative review, as opposed to a strictly defined systematic review [28]. This method works well for research in project management as well, provided that the relevant information is spread out across many managerial and technical fields [29].

2.1 Search Strategy and Study Selection

Google Scholar, ScienceDirect, IEEE Xplore, Web of Science, Scopus, and SpringerLink were all scanned for relevant material. Studies from earlier years were only kept when they demonstrated well-established ideas or methodologies that were critical to comprehending more recent studies; the primary search window was from 2020 to September 2026.

Software project management and performance, AI and ML, planning and resource optimization, and human and organizational variables were the four primary categories into which search phrases were grouped. Agile, DevOps, software project performance, software project management, project scheduling, effort estimate, machine learning, risk management, human factors, and decision support were common pairings.

Research was considered for inclusion if it provided insight into a management-related issue, such as estimating, planning, scheduling, resource utilization, risk, team performance, monitoring, or decision assistance for managers. When presenting methodologies that may be used to project management, such as optimization, intelligent monitoring, human-in-the-loop interaction, or automated performance assessment, software-intensive engineering studies were also taken into account. Exclusion criteria included research without a distinct methodological or managerial contribution, online sources that were not scholarly, and duplicate data.

2.2 Thematic Synthesis

Statistical synthesis of the chosen research was not possible due to the fact that the evaluated literature makes use of diverse datasets, aims, performance metrics, and experimental designs. Thematic synthesis was used instead for comparing the studies. Planning and scheduling, human factors in resource allocation, data-driven monitoring and prediction, and decision support for managers with the use of artificial intelligence were the four interrelated areas that were the focus of the investigation.

We carefully examined each study's management challenge, suggested solution, data or resource type, and stated performance result. Rather than focusing just on technical implementation, this framework enabled solutions from various software and engineering contexts to be compared based on their management purpose. Below, we apply the resultant synthesis to the optimization of software project performance by identifying common processes, constraints, and transferable methodologies.

3. Integrated Approaches for Software Project Performance Optimization

3.1 Resource Allocation and Human-Centered Decision Support

One of the key connections between project planning and real project performance is resource allocation. In distributed Agile development, managers need to think about team collaboration, communication, and location in addition to matching tasks with available capabilities. More consistent allocation choices may be supported by past project data when machine-learning algorithms are employed to forecast appropriate work assignments [30]. Because of the potential for delays, knowledge loss, and extra effort that may result from unexpected staff leave, employee availability is another factor that has to be taken into account throughout time. An approach to academic human resource management that utilizes interactive machine learning shown the ability to assess attrition risk and provide it to project managers in a user-friendly decision-support interface [31].

The scope of predictive human resource analysis is expanding beyond that of individual projects. To aid with retention choices, supervised machine-learning algorithms may sift through massive HR (Human Resources) databases for trends linked to voluntary employee turnover [32]. Operational choices might also make use of performance data. Transparent criteria, as opposed to just subjective management assessment, may be used to evaluate employee participation in a decision-support application based on job completion, role expectations, efficiency indicators, and performance gaps [33].

The efficiency of software delivery follows a similar logic. Despite the abundance of operational metrics produced by DevOps settings, effective management calls for a small collection of indications that can be understood and connected to improvement initiatives. Metrics and KPIs for DevOps and operational performance evaluation were uncovered and structured via a multivocal assessment [34]. Because model behavior might alter when forecasts go beyond the range of actual data, predictive information also needs cautious handling. The significance of choosing prediction techniques based on the available data and the intended forecasting job is highlighted by comparative study on regression models, interpolation, and extrapolation [35]. Predictions about future effort, duration, or resource consumption may go beyond previously observed project circumstances, which is a major concern in software project forecasting.

3.2 Scheduling, Risk, and Multi-Criteria Optimization

There are many moving parts in software projects, including conflicting goals, scarce resources, rival technologies, and technical dependencies. Scheduling is more than just creating a schedule; it's also an optimization challenge. In complicated real-world scheduling scenarios, where many

operational constraints need to be fulfilled concurrently, constraint programming and hybrid optimization approaches have proven useful [36]. Engineering applications that rely heavily on software use the same optimization philosophy. The performance of the resultant solution is affected by the choice of the underlying model, according to a graph-based robotic path-planning research that examined cylindrical, spherical, and Cartesian workspace representations [37]. This exemplifies a more general idea from the viewpoint of project management: the way an issue is represented may influence the accuracy and efficiency of the decision that is derived from it.

It is important to include risk prediction into this planning process. Software project risk assessment has made direct use of artificial neural networks with the use of past data on project complexity, financial status, performance, and schedule adherence [38]. Alternatives may vary in performance across many parameters, and managers may have to select between them. To systematically rank alternatives and weight criteria, hybrid multi-criteria approaches like BWM-PROMETHEE II are useful [39]. The underlying decision process may be applied to software project choices like technology selection, supplier evaluation, project prioritization, or resource-strategy comparison, even though it was shown in a material-recovery context.

What occurs after a plan is formed is also an important factor to optimize a project for. Because of the rapid obsolescence of prior decisions brought about by distributed components, frequent releases, and changing operational circumstances, continuous monitoring is essential in modern DevOps and microservice systems. Various capabilities, assumptions, and limits were found in a survey of DevOps and microservices monitoring technologies, which impact the selection of suitable monitoring solutions [40]. Equally amenable to dynamic treatment is the distribution of human resources. An example of how several management constraints may be integrated into a single allocation tool is a machine-learning program that was created for research projects. This application evaluates individuals according to their abilities, availability, workload, cost, past performance, and preferences [41].

3.3 Monitoring, Quality, and Early Detection

The success of a project is dependent on more than just its planning. In order to catch organizational or technological irregularities before they turn into costly breakdowns, monitoring is necessary. One application of this notion is the use of automated systems to identify defects. In the midst of an operational process, computer vision systems built on YOLO (You Only Look Once) architectures may detect quality flaws automatically and provide quick feedback [42]. In software infrastructure, the concept of continuous observation also emerges. Concurrent processing, Domain Name System (DNS) handling, proxy functionality, and traffic management are all aspects of a multithreaded Virtual Private Network (VPN) program that is built to monitor web traffic [43]. These examples show how important it is to think about runtime behavior, component coordination, and performance monitoring when developing software and managing a project.

When researching a topic, software repositories are a goldmine of knowledge. Everything from tasks and defects to actors and priorities as well as dates and times of work may be found in issue trackers and version control systems. Project status, actor actions, data quality, and process inefficiencies may all be measured with the use of these records thanks to multidimensional repository mining [44]. It is also possible to include automated quality monitoring into the workflows of engineering software. One use case for YOLO-v8 was the detection of 3D-printed component flaws; this case showed how an AI model may identify flaws caused by process parameter changes early on [45]. The management approach is the same whether you're working in software development or the application domain: automating problem identification as soon as it occurs shortens the time it takes to decide to fix an issue.

When considered together, these studies demonstrate that there is no silver bullet solution that can maximize software project performance. Allocating resources, making predictions, scheduling, assessing risks, monitoring, and implementing quality control are all interdependent processes that must be managed effectively. While optimization and AI may help with this process, they really shine when the results are easy to grasp and project managers can tweak them based on real-world circumstances.

4. Human-Centered and Adaptive Decision Support

4.1 Workload, Team Capacity, and Early Warning

Assigning tasks to the appropriate people and keeping workloads manageable throughout project execution are two of the most important factors influencing project performance. In order to aid in team allocation and staff retention choices, data-driven HR analysis may include metrics for performance, job satisfaction, training, workload, tenure, and attrition [46]. Software projects follow a similar pattern; managers there need to be alert for spikes in workload so they can prevent them from impacting delivery schedule, quality, or team stability.

Concentration throughout time is crucial as well. Digital event records may be aggregated into a pressure indicator and used to detect times of simultaneous activity, as shown by the Assessment Pressure Mapper [47]. Even though the program was originally developed for managing academic evaluations, the same principle may be used to software development projects. It is possible to identify times of increased workload and coordination risk by aggregating commit activity, issue resolution, testing events, sprint deadlines, and deployment data across time.

The trend toward software engineering with the user in mind is congruent with this concept. More than just data sources or targets of automated choices, engineers and users should be seen as active participants in the software ecosystem in AI-enabled software development, according to recent study [48]. What this implies for project management is that indications of workload should help managers take action without taking the place of contextual judgment.

4.2 Human-in-the-Loop Intelligent Systems

When software governs user-facing systems, human-centered project management takes on added significance. A software-intensive system may manage numerous technological modules while keeping continuous connection with the user [49]. An intelligent EMDR (Eye Movement Desensitization and Reprocessing) platform combines visual, haptic and auditory components with an AI chatbot. Like most complicated software projects, this one included a number of interdependent tasks, such as software development, device integration, testing, user interaction, and performance assessment.

Purpose, user values, trust, interaction, and deployment environment features are all interdependent on system performance, according to human-centered AI research [50]. Because technical optimization might provide an informally efficient solution that is hard to implement or accept, these considerations are important for software project planning.

The similar principle applies to an intelligent EMDR software created to help kids deal with mental health issues [51]. Instead of relying only on AI, the solution integrated software features with human assistance and an application-specific methodology. Projects like these show how important it is for managers of intelligent software systems to sync up the technical aspects of development with validation, usability, user needs, and the application's operational environment. The proceedings of the eLearning and Software for Education conferences independently record the publishing and its DOI.

4.3 Complex Software-Intensive Projects and Management Implications

System integration with sensors, software, signal processing, machine learning, and physical devices makes the management difficulty even more apparent. According to studies on BCIs, creating one is a complex interdisciplinary effort that calls for expertise in areas such as hardware, human interaction, signal processing, classification, interface design, and interface design [52]. Because of these interdependencies, BCI applications are good examples of software-intensive projects that need the coordination of several specialized resources and technical tasks.

One real-world example is a Raspberry Pi-based EEG (Electroencephalography) BCI prosthetic lower limb device [53]. Physical validation, software communication, embedded processing, data interpretation, motor control, and electroencephalogram (EEG) collection were all features of the prototype. This study is relevant from a project management standpoint not because of the medical application per se, but because of the need to integrate human contact, specialized skills, validation activities, and related software and hardware tasks into a single development process. This integrated architecture was tested in real-time and its implementation is detailed in the published article.

A more holistic view of software project success is backed by these research. Planning, allocating human resources, monitoring workloads, identifying technological dependencies, interacting with users, validating, and maintaining ongoing managerial control are all essential components of effective project management. Data-driven approaches may show when action is required, and optimization and AI techniques can come up with ideas or forecasts. However, project context, human restrictions, and operational goals should stay tied to final judgments.

As a result, project data is continually turned into managerial information via an integrated management cycle. The inputs to a project specify its activities, resources, limitations, risks, and anticipated outcomes. Analysis of these inputs is what predictive and optimization algorithms do to come up with estimations or alternatives. After that, human decision-makers evaluate the data and choose the best course of action. Plans may be modified in response to changes in workload, resources, hazards, or technical situations thanks to monitoring data that is returned to the system throughout execution. Instead of being a one-and-done planning exercise done at the start of a software project, optimization becomes an ongoing feedback loop in this manner.

5. Discussion and Conclusions

Instead of seeing software project performance as a collection of separate technical tasks, the literature suggests that it is better seen as a related management challenge. Integrating projects, managing scope, communicating, analyzing risks, and making choices involving stakeholders are all areas that may benefit from artificial intelligence. However, the actual effectiveness of AI relies on the information that is accessible and how managers perceive its outputs [54]. On the software engineering level, AI may aid in requirements analysis, coding, testing, prediction, and monitoring; nevertheless, there are still significant obstacles to overcome, such as explainability, privacy, bias, and integration with current development methods [55].

The quality of the plan is still paramount. Problems with precise work estimates in Agile development may arise from a variety of sources, including inadequate data, team dynamics, estimating methods, managerial choices, and external business factors [56]. Technical indicators and delivery speed are not sufficient to assess project effectiveness since human factors significantly impact the relationship between software engineers and end users [57]. In this review, we found that automated suggestions help managers make choices, but they shouldn't be seen as a replacement for human judgment and contextual knowledge. Our results support this person-in-the-loop approach.

The importance of both data quality and system integration cannot be overstated. Project uncertainty is most often caused by issues with data availability and quality while working on software projects that rely on machine learning [58]. There is a lot of variation in language, implementation models, and software solutions among project management information systems in organizational research, which suggests that project-support tools frequently need to be adapted to fit the specific environment of the organization [59]. Simultaneously, big language models are adding intelligent assistance to a wide range of software engineering tasks; nevertheless, there is still a need for study into how to assess these models, ensure their reliability, and integrate them appropriately into development processes [60].

A number of restrictions apply to this evaluation. It employs an integrative approach instead of a statistical review methodology, and the research that were considered vary in terms of datasets, fields of application, performance metrics, and experimental settings. Similarities in software project management and software-intensive engineering applications in the areas of planning, resource coordination, monitoring, and decision support led to the transfer of certain evidence from the former to the latter. Validation of the hypothesized correlations should be conducted using genuine software project data, such as issue repositories, sprint histories, records of resource allocation, workload indicators, defects, and delivery results, in future study.

When looking at software project performance optimization, the data shows that no one technique or management strategy is enough. Planning, allocating resources, doing predictive analyzes, monitoring risks, considering human aspects, and maintaining management control are all better done as part of a continuous feedback loop. Human supervision is still required for understanding limitations, assessing trade-offs, and adjusting choices to changing project circumstances, even if optimization algorithms and artificial intelligence may improve the quantity and quality of data accessible to managers. Future models and approaches that attempt to improve software development project performance may be grounded on this integrated viewpoint.

References

1. Nenni, M. E., De Felice, F., De Luca, C., & Forcina, A. (2025). How artificial intelligence will transform project management in the age of digitization: A systematic literature review. *Management Review Quarterly*, 75, 1669–1716. <https://doi.org/10.1007/s11301-024-00418-z>
2. Buzducea, C.-A., Drăgoi, M.-V., Cristoiu, C., Puiu, R.-A., Puiu, M., Petrea, G., & Navligu, B.-C. (2026). Machine learning in education: Predicting student performance and guiding institutional decisions. *Education Sciences*, 16(1), 76. <https://doi.org/10.3390/educsci16010076>
3. Salimimoghadam, S., Ghanbaripour, A. N., Tumpa, R. J., Rahimi, A. K., Golmoradi, M., Rashidian, S., & Skitmore, M. (2025). The rise of artificial intelligence in project management: A systematic literature review of current opportunities, enablers, and barriers. *Buildings*, 15(7), 1130. <https://doi.org/10.3390/buildings15071130>
4. Drăgoi, M.-V., Puiu, R.-A., Petrea, G., Cristoiu, C. A., & Dumitrescu, C.-I. (2026). Willingness to allow educational data use for learning analytics in higher education: Trust and governance predictors: An exploratory study. *Education Sciences*, 16(4), 637. <https://doi.org/10.3390/educsci16040637>
5. Su, X., & Ayob, A. H. (2025). Artificial intelligence in project success: A systematic literature review. *Information*, 16(8), 682. <https://doi.org/10.3390/info16080682>
6. Drăgoi, M.-V., Puiu, R.-A., Petrea, G., Burtea, C., Spiridon-Mocioacă, T.-M., Puiu, M., & Suciu, C. P. (2025). Improving the learning process for students. Personalizing learning through software applications. *U.P.B. Scientific Bulletin, Series C*, 87(1), 157–166. https://www.scientificbulletin.upb.ro/rev_docs_arhiva/full465_531870.pdf
7. Jørgensen, M. (2024). A systematic literature review on characteristics of the front-end phase of agile software development projects and their connections to project success. *Journal of Systems and Software*, 216, 112155. <https://doi.org/10.1016/j.jss.2024.112155>
8. Drăgoi, M.-V., Nisipeanu, I., Puiu, R.-A., Tache, F.-G., Spiridon-Mocioacă, T.-M., Hank, A., & Cristoiu, C. (2025). An inclusive offline learning platform integrating gesture recognition and local AI models. *Biomimetics*, 10(10), 693. <https://doi.org/10.3390/biomimetics10100693>
9. Amaro, R., Pereira, R., & da Silva, M. M. (2025). Mapping DevOps capabilities to the software life cycle: A systematic literature review. *Information and Software Technology*, 177, 107583. <https://doi.org/10.1016/j.infsof.2024.107583>
10. Drăgoi, M.-V., Cristoiu, C. A., Nechita, R.-M., Navligu, B.-C., & Verdete, B.-M. (2026). A Python-based framework for learning-from-demonstration in robotic object sorting: Comparative evaluation of lightweight classifiers. *Applied Sciences*, 16(10), 5107. <https://doi.org/10.3390/app16105107>
11. Terlapu, P. V., Raju, K. K., Kumar, G. K., Rao, G. J., Kavitha, K., & Samreen, S. (2024). Improved software effort estimation through machine learning: Challenges, applications, and feature importance analysis. *IEEE Access*, 12, 138663–138701. <https://doi.org/10.1109/ACCESS.2024.3457771>
12. Cristoiu, C. A., Drăgoi, M.-V., Nechita, R.-M., Verdete, B.-M., Dudici, C. L., & Cusma, C. N. (2026). Python software application for obstacle-avoiding path planning in RoboDK using free space graph and robot level validation. *Applied Sciences*, 16(10), 4786. <https://doi.org/10.3390/app16104786>
13. Masso, J., Pino, F. J., Pardo, C., García, F., & Piattini, M. (2020). Risk management in the software life cycle: A systematic literature review. *Computer Standards & Interfaces*, 71, 103431. <https://doi.org/10.1016/j.csi.2020.103431>
14. Drăgoi, M.-V., Nisipeanu, I., Frimu, A., Tălîngă, A.-M., Hadăr, A., Dobrescu, T. G., Suciu, C. P., & Manea, A. R. (2024). Real-time home automation system using BCI technology. *Biomimetics*, 9(10), 594. <https://doi.org/10.3390/biomimetics9100594>
15. Salido O., M. G., Borrego, G., Palacio Cinco, R. R., & Rodríguez, L.-F. (2023). Agile software engineers' affective states, their performance and software quality: A systematic mapping review. *Journal of Systems and Software*, 204, 111800. <https://doi.org/10.1016/j.jss.2023.111800>
16. Drăgoi, M.-V., Frimu, A.-V., Postelnicu, A., Puiu, R.-A., Petrea, G., & Hank, A. (2026). Interactive teleoperation of an articulated robotic arm using vision-based human hand tracking. *Biomimetics*, 11(2), 151. <https://doi.org/10.3390/biomimetics11020151>

17. Godliauskas, P., & Šmite, D. (2025). The well-being of software engineers: A systematic literature review and a theory. *Empirical Software Engineering*, 30, 35. <https://doi.org/10.1007/s10664-024-10543-8>
18. Cristoiu, C. A., Drăgoi, M.-V., Cojocaru, A., & Spănu, P. (2026). A vision-based software safety monitoring tool for operators in RoboDK robotic cells: A simulation-based proof-of-concept study using workspace masks and image processing. *Technologies*, 14(6), 373. <https://doi.org/10.3390/technologies14060373>
19. Artigues, C., Hartmann, S., & Vanhoucke, M. (2026). Fifty years of research on resource-constrained project scheduling explored from different perspectives. *European Journal of Operational Research*, 328(2), 367–389. <https://doi.org/10.1016/j.ejor.2025.03.024>
20. Cristoiu, C. A., Drăgoi, M.-V., Nechita, R.-M., Stoian, B. N., Dudici, C. L., Puiu, R.-A., & Petrea, G. (2026). The impact of geometric continuities (C1, C2, and C3) on the trajectories of industrial robots. *Technologies*, 14(3), 176. <https://doi.org/10.3390/technologies14030176>
21. van der Beek, T., van Essen, J. T., Pruyn, J., & Aardal, K. (2025). Exact solution methods for the resource constrained project scheduling problem with a flexible project structure. *European Journal of Operational Research*, 322(1), 56–69. <https://doi.org/10.1016/j.ejor.2024.10.029>
22. Cristoiu, C. A., Drăgoi, M.-V., Ivan, A. M., Nechita, R.-M., Grecu, I., Puiu, R.-A., Petrea, G., & Popescu, E. (2025). Optimization method based on the minimum action principle for trajectory length of articulated manipulators. *Technologies*, 13(11), 490. <https://doi.org/10.3390/technologies13110490>
23. Mohammad, A., & Chirchir, B. (2024). Challenges of integrating artificial intelligence in software project planning: A systematic literature review. *Digital*, 4(3), 555–571. <https://doi.org/10.3390/digital4030028>
24. Drăgoi, M.-V., Nisipeanu, I., Marin, I., Cristoiu, C. A., & Alexe, C.-G. (2026). Bio-inspired gaze and neural command fusion for assistive smartphone interaction. *Biomimetics*, 11(7), 514. <https://doi.org/10.3390/biomimetics11070514>
25. Mahdi, M. N., Mohamed Zabil, M. H., Ahmad, A. R., Ismail, R., Yusoff, Y., Cheng, L. K., Mohd Azmi, M. S. B., Natiq, H., & Happala Naidu, H. (2021). Software project management using machine learning technique—A review. *Applied Sciences*, 11(11), 5183. <https://doi.org/10.3390/app11115183>
26. Drăgoi, M.-V., Hadăr, A., Goga, N., Baci, F., Ștefan, A., Grigore, L. Ș., Gorgoteanu, D., Molder, C., & Oncioiu, I. (2023). Contributions to the dynamic regime behavior of a bionic leg prosthesis. *Biomimetics*, 8(5), 414. <https://doi.org/10.3390/biomimetics8050414>
27. Meckenstock, J.-N. (2024). Shedding light on the dark side—A systematic literature review of the issues in agile software development methodology use. *Journal of Systems and Software*, 211, 111966. <https://doi.org/10.1016/j.jss.2024.111966>
28. Oermann, M. H., & Knafl, K. A. (2021). Strategies for completing a successful integrative review. *Nurse Author & Editor*, 31(3–4), 65–68. <https://doi.org/10.1111/nae2.30>
29. McLeod, S. (2021). Feasibility studies for novel and complex projects: Principles synthesised through an integrative review. *Project Leadership and Society*, 2, 100022. <https://doi.org/10.1016/j.plas.2021.100022>
30. Al-Fraihat, D., Sharrab, Y., Al-Ghuwairi, A.-R., Alzabut, H., Beshara, M., & Algarni, A. (2024). Utilizing machine learning algorithms for task allocation in distributed agile software development. *Heliyon*, 10(21), e39926. <https://doi.org/10.1016/j.heliyon.2024.e39926>
31. Puiu, R.-A., Drăgoi, M.-V., Tache, F.-G., Spiridon-Mocioacă, T.-M., Petrea, G., & Hank, A. (2026). Interactive attrition prediction for academic human resources management using machine learning. *U.P.B. Scientific Bulletin, Series C*, 88(2), 217–232. https://www.scientificbulletin.upb.ro/static/pdfs/a2026nr2c_16_16653_Roxana_Puiu_2_pag.pdf
32. Veglio, V., Romanello, R., & Pedersen, T. (2025). Employee turnover in multinational corporations: A supervised machine learning approach. *Review of Managerial Science*, 19, 687–728. <https://doi.org/10.1007/s11846-024-00769-7>
33. Tache, F.-G., Drăgoi, M.-V., Puiu, R.-A., Petrea, G., & Pascaru, R.-C. (2025). A decision-support tool for evaluating employee contribution toward organizational excellence. *Proceedings of the 12th International Conference on Management and Industrial Engineering (ICMIE 2025)*, Paper 109. <https://doi.org/10.56177/12icmie2025.109>
34. Amaro, R., Pereira, R., & Mira da Silva, M. (2024). DevOps metrics and KPIs: A multivocal literature review. *ACM Computing Surveys*, 56(9), Article 231. <https://doi.org/10.1145/3652508>
35. Drăgoi, M.-V., Afram, R. M., Alzubaidi, A. A. A. M. H., Jiga, G. G., Ali, H. A., Puiu, R.-A., Petrea, G., & Hank, A. (2026). Comparative analysis of interpolation and extrapolation methods in regression models. *U.P.B. Scientific Bulletin, Series C*, 88(1), 145–160. <https://www.scientificbulletin.upb.ro/seria/c/arhiva?year=2026>
36. Geibinger, T., Mischek, F., & Musliu, N. (2024). Investigating constraint programming and hybrid methods for real world industrial test laboratory scheduling. *Journal of Scheduling*, 27, 607–622. <https://doi.org/10.1007/s10951-024-00821-0>
37. Cristoiu, C.-A., Drăgoi, M.-V., & Georgescu, V.-C. (2026). Comparative analysis of Cartesian, cylindrical and spherical grids in a graph-based obstacle-avoidance planner for industrial robots. *Applied Sciences*, 16(12), 6189. <https://doi.org/10.3390/app16126189>

38. Alatawi, M. N., Alyahyan, S., Hussain, S., Alshammari, A., Aldaej, A. A., Alali, I. K., & Alwageed, H. S. (2023). A data-driven artificial neural network approach to software project risk assessment. *IET Software*, 2023, 4324783. <https://doi.org/10.1049/2023/4324783>
39. Nechita, R.-M., Deselnicu, D.-C., Băjenaru, V.-D., Istrițeanu, S.-E., Cristoiu, C., & Drăgoi, M.-V. (2026). Optimizing material recovery from photovoltaic waste: A performance ranking using hybrid BWM-PROMETHEE II. *Sustainability*, 18(8), 3750. <https://doi.org/10.3390/su18083750>
40. Giamattei, L., Guerriero, A., Pietrantuono, R., Russo, S., Malavolta, I., Islam, T., Dînga, M., Koziolek, A., Singh, S., Armbruster, M., Gutierrez-Martinez, J. M., Caro-Alvaro, S., Rodriguez, D., Weber, S., Henss, J., Fernandez Vogelin, E., & Simon Panojo, F. (2024). Monitoring tools for DevOps and microservices: A systematic grey literature review. *Journal of Systems and Software*, 208, 111906. <https://doi.org/10.1016/j.jss.2023.111906>
41. Tache, F.-G., Drăgoi, M.-V., Puiu, R.-A., Petrea, G., & Pascaru, R.-C. (2026). Application in research projects on the innovation found in the field of engineering of polymer and composites products. *Macromolecular Symposia*, 415(3), e70379. <https://doi.org/10.1002/masy.70379>
42. Liu, J., Cai, Z., He, K., Huang, C., Lin, X., Liu, Z., Li, Z., & Chen, M. (2024). An efficient printing defect detection based on YOLOv5-DCN-LSK. *Sensors*, 24(23), 7429. <https://doi.org/10.3390/s24237429>
43. Drăgoi, M.-V., Maier, A. N., Alzubaidi, A. A. M. H., Suciu, C. P., Ali, H. A., Dobrescu, T. G., Hadăr, A., Puiu, R.-A., Petrea, G., & Avăsiloaie, A. P. (2025). Multithreading VPN application to monitor web traffic and unblock geoblocking websites. *U.P.B. Scientific Bulletin, Series C*, 87(4). https://www.scientificbulletin.upb.ro/static/pdfs/rez17b_486740.pdf
44. Reszka, Ł., Sosnowski, J., & Dobrzyński, B. (2023). Enhancing software project monitoring with multidimensional data repository mining. *Electronics*, 12(18), 3774. <https://doi.org/10.3390/electronics12183774>
45. Tălîngă, A.-M., Hadăr, A., Drăgoi, M.-V., Nisipeanu, I., Ali, H. A., & Suciu, C. P. (2024). YOLO-v8 in capturing imperfections generated by changing 3D printer parameters. *U.P.B. Scientific Bulletin, Series C*, 86(4), 253–266. https://www.scientificbulletin.upb.ro/rev_docs_arhiva/full72b_799902.pdf
46. Buzducea, C.-A., Puiu, M., Tache, F.-G., & Petrea, G. (2025). Data-driven optimization of research team human resources. *FAIMA Business & Management Journal*, 13(4), 130–138. https://faimajournal.upb.ro/archive/Revista_FAIMA_Vol13_Issue4_Dec2025.pdf
47. Drăgoi, M., Buzducea, C., Marin, A., & Boanță, L. (2026). Assessment pressure mapper for visualizing peaks in digital evaluation load. *FAIMA Business & Management Journal*, 14(1), 34–42. https://faimajournal.upb.ro/archive/RevistaFAIMA_Martie2026.pdf
48. Abrahão, S., Grundy, J., Pezzè, M., Storey, M.-A., & Tamburri, D. A. (2025). Software engineering by and for humans in an AI era. *ACM Transactions on Software Engineering and Methodology*, 34(5), Article 129, 1–46. <https://doi.org/10.1145/3715111>
49. Goga, N., Boiangiu, C.-A., Vasilateanu, A., Popovici, A.-F., Drăgoi, M.-V., Popovici, R., Gancea, I. O., Pîrlog, M. C., Popa, R. C., & Hadăr, A. (2022). An efficient system for eye movement desensitization and reprocessing (EMDR) therapy: A pilot study. *Healthcare*, 10(1), 133. <https://doi.org/10.3390/healthcare10010133>
50. Schmager, S., Pappas, I. O., & Vassilakopoulou, P. (2025). Understanding human-centred AI: A review of its defining elements and a research agenda. *Behaviour & Information Technology*, 44(15), 3771–3810. <https://doi.org/10.1080/0144929X.2024.2448719>
51. Gancea, I. O., Drăgoi, M.-V., Popovici, A.-F., Goga, N., Dragomir, R., Goga, M., & Dinu, M. C. (2020). An intelligent EMDR system for helping students with psychological problems. *eLearning & Software for Education*, 2, 207–214. <https://doi.org/10.12753/2066-026X-20-111>
52. Karikari, E., & Koshechkin, K. A. (2023). Review on brain-computer interface technologies in healthcare. *Biophysical Reviews*, 15(5), 1351–1358. <https://doi.org/10.1007/s12551-023-01138-6>
53. Drăgoi, M.-V., Hadăr, A., Goga, N., Grigore, L.-Ș., Ștefan, A., & Ali, H. A. (2023). Design and implementation of an EEG-based BCI prosthetic lower limb using Raspberry Pi 4. *U.P.B. Scientific Bulletin, Series C*, 85(3), 353–366. https://www.scientificbulletin.upb.ro/rev_docs_arhiva/full704_649223.pdf
54. Almeida, P. M., Fernandes, G., & Santos, J. M. R. C. A. (2025). Artificial intelligence tools for project management: A knowledge-based perspective. *Project Leadership and Society*, 6, 100196. <https://doi.org/10.1016/j.plas.2025.100196>
55. Abbas, T., Rathore, S. A., Turki, A., Khan, S., Alghushairy, O., & Daud, A. (2025). Enhancing software engineering with AI: Innovations, challenges, and future directions. *IET Software*, 2025(1), 5691460. <https://doi.org/10.1049/sfw2/5691460>
56. Pasuksmit, J., Thongtanunam, P., & Karunasekera, S. (2024). A systematic literature review on reasons and approaches for accurate effort estimations in Agile. *ACM Computing Surveys*, 56(11), Article 269, 1–37. <https://doi.org/10.1145/3663365>
57. Gunatilake, H., Grundy, J., Hoda, R., & Mueller, I. (2024). The impact of human aspects on the interactions between software developers and end-users in software engineering: A systematic literature review. *Information and Software Technology*, 173, 107489. <https://doi.org/10.1016/j.infsof.2024.107489>

-
58. Lin, C.-T., & Huang, S.-J. (2024). Technical risk model of machine learning based software project development: A multinational empirical study using modified Delphi-AHP method. *Information and Software Technology*, 171, 107449. <https://doi.org/10.1016/j.infsof.2024.107449>
 59. Rebuglio, M., Zenezini, G., Ottaviani, F. M., & De Marco, A. (2025). Project management information systems: A systematic review. *Procedia Computer Science*, 256, 1739–1747. <https://doi.org/10.1016/j.procs.2025.02.313>
 60. Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8), Article 220, 1–79. <https://doi.org/10.1145/3695988>