International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.141, September 2026

Data-driven Methodology for Resource Planning in Large-scale Software Development

Piotr Grobelny
University College of Professional Education
Wrocław, Poland

ABSTRACT

Large-scale software development organizations operate in environments characterized by multiple parallel value streams, distributed teams, and dependencies on external stakeholders. Although agile software development enables flexibility, existing project management approaches often provide insufficient support for long-term resource planning across complex portfolios. This paper proposes and validates a data-driven resource planning methodology that bridges project management and agile product development while preserving agile principles and improving planning predictability. The methodology integrates four complementary elements: net capacity estimation, historical data analysis, planning policies, and a planning matrix for allocating resources across multiple value streams. Historical data extracted from work management systems such as Jira are continuously analyzed to estimate task effort and delivery timelines, enabling iterative calibration of planning parameters. The methodology was developed and evaluated through its implementation in a large-scale industrial software development organization. The evaluation demonstrated improved planning accuracy, more effective utilization of development capacity, better coordination of parallel projects, and greater transparency for stakeholders. Continuous refinement of planning parameters based on empirical project data reduced discrepancies between planned and actual workloads while supporting more predictable delivery planning. The results indicate that the proposed methodology provides a practical and scalable framework for resource planning in complex software development environments and establishes a foundation for future research on integrating predictive analytics and intelligent decision-support mechanisms to further enhance planning accuracy and resource allocation.

Keywords

data-driven resource planning, agile project management, large-scale software development

1. INTRODUCTION

In the software development space, there are numerous project management as well as product development management methodologies. Each approaches task and resource planning based on a specific perspective and focuses on planned projects or product functionality. According to the author of this article, there is a need to develop a planning approach that combines these two perspectives. He proposed a methodology based on observations and analysis of a specific case study developed and implemented at a software development company. Therefore, this method is not universal, but it can certainly be applied to companies operating in environments similar to the one described later in this article.

The aforementioned planning methodology was developed in real-world conditions, where the author was unable to directly apply the most popular development team planning methods. The resulting concept was subjected to observation, which led to the creation and fine-tuning of a programmer's work planning model. The main limitations compared to existing approaches that initiated work on the new model were the inflow of work from multiple value streams and the management of multicultural development teams, whose work was preceded by requirements analysis conducted by analytical teams. Furthermore, the work depended on two types of third parties: clients and companies providing industry solutions. The software being developed was a transactional platform that connected these parties. This characteristic situation caused interruptions in programmers' work, requiring coordination between the two parties and preventing full agile work. Another challenge was scheduling work to efficiently utilize resources while awaiting decisions from both parties.

The planning methodology proposed in this article is not an attempt to replace agile methodologies in software development, but it can be very helpful in the situation described above, thanks to the fact that it maintains the culture of working in an agile manner and at the same time allows for the elimination of the mentioned inconveniences.

Resource planning in this methodology is data-driven. The author will demonstrate the specific principles of this method, as well as how to fine-tune the planning model, which will be documented with the results of implementing the method in a real-world environment.

Problem statement

Current trends in automation and scalability have paved the way for the development of software as internet platforms. On the one hand, this software is characterized by deep integration with other systems via web services within so-called virtual organizations, where information exchange occurs without human intervention in a machine-to-machine communication scheme. On the other hand, platforms enable the development of diverse solutions for end customers with specific requirements and markets. Using one

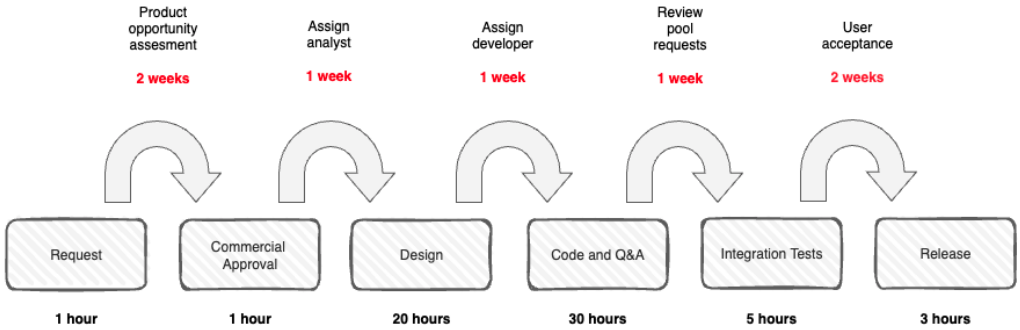

Fig. 1. Identification of wasting time within the development process

central platform to build dedicated solutions makes it easier to scale the business that this platform represents as a whole [8].
Considering the above, it can be seen that dedicated solutions built on a central platform generate various value streams resulting from the needs of end customers. This creates a need to manage the platform development effort through effective resource planning, taking into account the specific requirements of each value stream. Therefore, the author of this article set himself the goal of developing a dedicated software platform development methodology while maintaining the agile work culture defined in the Agile Manifesto and taking into account the best practices from methodologies such as Scrum, Kanban, SAFe, Nexus, Agile Project and Program Management [2, 4].
The presented methodology is an approach to effective and reasonable resource planning when delivering large-scale projects related to the development of a software platform by multiple product, analytical and development teams working simultaneously in an environment of constant change. From an economic perspective, planning the work of platform development resources should take into account the following aspects: delivering value frequently and incrementally, and delivering solutions to the market as quickly as possible, which in turn accelerates the emergence of new cash flows in the business. Therefore, one of the most important aspects of good planning is reducing non-value-added waste in the software development process. Many real-world processes encounter delays, which delay software delivery. This means that reducing delays is the fastest way to minimize time to market.
Schema presented in Figure 1 does not represent a standard process, but rather a concept for identifying wasting time. Each organization has a process tailored to its needs, often based on continuous delivery or continuous deployment, and may differ from the one presented by the author. Most program management, project management, and agile software product development methodologies focus on minimizing wasted time identified through process mapping techniques [12]. As shown in Figure 1, the 60 hours of effort required to deliver a certain amount of value will take seven weeks. The resulting process has a flow efficiency of 21%, calculated as the ratio of active working time (7.5 working days) to the total process lead time (35 working days). Consequently, there is a need for a planning methodology that systematically reduces waiting time while enabling more effective resource management in large-scale software development projects, where multiple teams, shared specialists, and task dependencies significantly increase planning complexity.
What should be done if, for objective reasons, wasted time cannot be minimized? For example, the platform being developed integrates services provided by various third parties within a virtual organization for end customers whose requirements also differ. Collaboration with all stakeholders means that responses are typically provided within one or two weeks.
In such a situation, how can the work of platform development resources be optimally planned without the benefits of agile work being lost sight of? The methodology presented in this article addresses this problem. It was created by the author and was implemented on a pilot basis in the company's department responsible for software platform development. Subsequently, following a positive evaluation, it was implemented on a larger scale within the organization. Its foundations and implementation results are discussed in the following chapters.

Research objective

The research problem addressed in this study was the development of a methodology for predictable software development planning based on empirical observations of large-scale software engineering projects. The objective was not only to propose the methodology but also to evaluate, validate, and demonstrate its practical effectiveness under real organizational conditions. Consequently, the methodology was derived from empirical evidence collected during software development processes and subsequently assessed to verify that it improves planning accuracy, resource utilization, and delivery predictability.
The pillars that formed the basis for building the methodology were identified during the observation and analysis of software development processes in a large-scale organization. They represent the key factors that were found to have the greatest impact on planning accuracy, delivery predictability, and the overall effectiveness of development teams. Together, these pillars create a balanced framework that combines objective decision-making with efficient resource management while maintaining sustainable working conditions for development teams. The key pillars of the proposed methodology are as follows:

—Data-driven decision-making [7, 6]: The observation process involved collecting empirical data from software development projects and continuously tuning planning parameters based on historical performance. Instead of relying solely on expert judgment or subjective estimates, planning decisions are supported by objective evidence such as historical workload, elapsed time, and delivery performance. This enables the

planning model to evolve over time and adapt to changes in the organization and development process.

—Efficient utilization of capacity [9]: Software development organizations frequently execute multiple projects and value streams simultaneously while sharing the same development resources. Planning the work of several teams in parallel allows available capacity to be allocated more effectively, improves responsiveness to changing priorities, reduces idle time caused by external dependencies, and increases the organization's ability to meet business deadlines without compromising development quality.

—Ensuring the psychological safety of development team members [1]: Modern software development environments are characterized by continuously changing priorities, evolving customer requirements, and frequent communication with multiple stakeholders. Constant context switching and uncertainty can negatively affect both productivity and team well-being. The proposed methodology minimizes these effects by providing transparent planning rules, realistic workload allocation based on actual capacity, and clear communication of priorities. As a result, development teams can focus on delivering value instead of reacting to constant planning changes, supporting both sustainable development practices and long-term team effectiveness.

2. RELATED WORK

This chapter describes the most important aspects of software development methodologies that were drawn from and used to build the methodology described in this article. Agile project management, particularly through Scrum and Kanban methodologies, is widely used in environments characterized by volatility, uncertainty, complexity, and ambiguity. These methods prioritize active client participation and adaptive scope definition through continuous feedback and experimentation. Despite their common foundation, the Agile Manifesto, both methodologies operate effectively in slightly different environments [11]. Scrum is best suited for small organizations, research projects, startups, and wherever the final shape of the product is uncertain. Kanban, on the other hand, helps manage software maintenance projects, as well as wherever software elements are repeatable and typically delivered within a clearly defined process. For example, Scrum uses short, iterative sprints (2-4 weeks) for incremental product development, focusing on transparency, inspection, and adaptation through events, artifacts, and roles. Kanban, on the other hand, balances supply and demand for continuous workflow through visualization, Work in Progress (WIP) limits, flow management, explicit policies, feedback, and continuous improvement [10]. In essence, Scrum provides a structured, iterative approach with clear roles and ceremonies, aiming for predictable delivery within fixed timeboxes. Kanban, on the other hand, offers a more fluid and continuous approach focused on optimizing workflow and adapting to changes without strict timeboxes or roles [18]. Many organizations use a combined approach called Scrumban. While both methods offer distinct advantages, it is advisable to use them in combination. A hybrid approach of Kanban and Scrum can help balance structured planning with operational flexibility, allowing for rapid responses to market changes. This combination integrates Scrum's strategic planning with Kanban's continuous workflow, enhancing transparency and control over processes. Integrating Scrum and Kanban offers visual monitoring and transparency but doesn't resolve workflow constraints [11].

Recent studies also indicate that no single Agile methodology is universally optimal. Instead, software organizations should select and combine Agile practices according to project characteristics, organizational context, and business objectives [5].

Despite its numerous advantages, Agile adoption presents challenges such as resistance to change, the need for cultural shifts, and difficulties in scaling Agile practices in large, distributed teams. Effective implementation requires a strategic approach that includes selecting appropriate tools and techniques, balancing quality, cost, and schedule, adapting to organizational and project context, continuous stakeholder involvement and feedback, and emphasizing communication and collaboration [3, 5].

The proposed methodology also addresses all mentioned challenges. It was designed and implemented based on work with multiple, distributed, multicultural development teams, taking into account the economic needs of a global organization when implementing large software projects.

The literature describes many aspects of large-scale software development methodologies. Since Agile was designed for smaller teams, scaling it to large projects requires specific frameworks. These include the Scaled Agile Framework (SAFe), Agile Project Management (AgPM), and Nexus [14]. For successful large-scale Agile, it's crucial to manage communication, coordination, and alignment across numerous teams. For instance, organize regular cross-team meetings, workshops, and program increment planning sessions to foster collaboration and alignment, utilize visual management tools like Kanban boards or program boards to enhance transparency and track progress across teams [15].

Additionally, Gómez Sánchez et al. [9] provide a comprehensive survey of the Resource-Constrained Multi-Project Scheduling Problem (RCMPSP), summarizing optimization techniques for allocating limited resources across multiple concurrent projects. Although these approaches address scheduling and resource optimization, they are primarily algorithmic and do not propose a practical planning methodology tailored to large-scale software development organizations operating in Agile environments.

The methodology proposed in this paper complements this body of research by integrating empirical historical project data, Net Capacity estimation, Planning Policy, and Planning Matrix into a practical data-driven framework for software engineering resource planning. In this way, the proposed approach supports adaptive planning based on high-level project timelines and roadmaps. It enhances predictability for stakeholders in a dynamic market environment while enabling effective priority management in response to evolving business needs.

3. METHOD

The development of the principles was based on a characteristic feature of the presented methodology. Namely, it bridges the gap between the project approach, which relies on project management, and the agile approach, which relies on product development management. The proposed methodology can be said to bridge the gap between both worlds, reinforcing their complementarity. The principles are based on the collaboration of project managers representing different customer value streams and agile development teams that develop a product that meets the client's requirements.

The methodology allows project managers to build roadmaps and plans for specific business value streams and also allows product owners and development teams to appropriately build and prioritize backlogs. It provides project estimates regarding workload and expected delivery times for individual deliverables, while also

Table 1. Example of determining Net Capacity

January													
Team	**FTE**	**Work Days**	**Gross Capacity [MD]**	**Holidays**	**Illness**	**Trainings**	**ALM**	**Meetings**	**Bugs**	**Available Capacity [MD]**	**Team Efficiency**	**External Factors**	**Net Capacity [MD]**
X	5.9	20.0	118.0	1.8	2.95	5.9	5.0	2.5	9.4	103.6	90.0%	10.0%	82.9
Y	4.0	20.0	80.0	1.2	2.0	4.0	6.0	3.0	7.4	58.4	90.0%	10.0%	46.7
Z	4.6	20.0	92.0	1.4	2.3	4.6	3.9	3.2	8.3	68.3	90.0%	10.0%	54.6
Total Gross Capacity [MD]:			**290.0**	**Total Available Capacity [MD]:**						**230.3**	**Total Net Capacity [MD]:**		**184.2**

ensuring that development teams adhere to Agile principles during software development.

Net Capacity

The first principle of the method is determining Net Capacity. This defines the actual effort that teams can devote to specific work in a given period. The calculation model may vary across organizations, but the key is to ultimately determine the net effort that all teams can devote to analytical, programming, and QA work in a given month. Let's consider the Net Capacity model shown in Table 1 as an example.

Total Gross Capacity represents the total available working capacity of all software development employees in a given month, expressed as the product of the number of Full-Time Equivalents (FTEs) and the available working days (Man-Days, MD). From this value, elements that impact daily work, such as vacations, illnesses, training, meetings, and application lifecycle management (ALM) activities, were subtracted. The effort required to correct errors was also allocated within the model. As mentioned, the model components may vary across organizations; data may also be collected manually, retrieved from company systems such as time and attendance systems, or determined heuristically based on experience. Taking these elements into account, Available Capacity was determined. However, another factor that must be considered, which influences this value, is team effectiveness, which may vary due to the fact that teams are composed of individuals with different competencies and experience, as well as due to unpredictable events (external factors). By using these additional metrics, Net Capacity was obtained, which is defined as the basic effort that can be devoted strictly to analytical, programming, and testing tasks. Calculations show that, in reality, about 65% of working time can be devoted to specific work.

However, the most important thing for this principle is that the model must be agreed and approved by the employees belonging to the teams because it is, in a sense, a commitment to deliver value to the product being developed in a given month.

Data-Driven Approach

A Data-Driven Approach is a key element in adapting an organization's work planning model. It divides ongoing tasks into different task types within defined value streams. For each of these types, actual, average data from work tracking systems such as Jira is used. Based on the collected data, it is possible to more precisely estimate task completion times, allocate resources in the backlog, and modify planning policies. As a result, the planning model is gradually calibrated based on the team's actual work performance, becoming increasingly accurate and reliable over time.

Research by [13] confirms the above claims, demonstrating that accurate effort estimation is crucial for project planning, enabling schedule prediction, resource allocation, and the determination of the required project team. At the same time, estimation methods are empirically analyzed based on data from actual projects, which aligns with a data-driven approach to work management. The research findings also indicate that planning methods can be iteratively improved by analyzing their accuracy and effectiveness, leading to more effective planning models in Agile environments.

In the implementation described in this study, the primary source of historical data was the Jira work management system, where all development tasks were registered and classified according to predefined task categories, such as New Unit, Unit Extension, Feature, Large Bug Fix, Small Bug Fix, Technical Improvement. Time tracking information was collected using the Jira Tempo plugin, which records the actual effort spent on individual tasks by development teams. By combining task classification with recorded working time and workflow history, it was possible to calculate the average effort and elapsed time for each task type. These empirical metrics formed the basis for constructing the Planning Policy and continuously calibrating its parameters throughout the implementation. However, the proposed methodology is not dependent on Jira or Tempo. Any corporate workflow management ecosystem capable of recording task lifecycles, classifications, and actual work effort may serve as a source of historical data. Consequently, the methodology can be integrated with various enterprise project and workflow management systems, provided that they capture sufficient operational data to support data-driven planning.

Planning Policy

A Planning Policy formally represents the current resource allocation assumptions for the work process. This policy is based on values calculated based on empirical data from work tracking systems such as Jira. Two key metrics are used in particular: workload in terms of the average effort required to complete specific task types and the elapsed time, measured from initiation to delivery. Using these metrics enables more rational work planning, aligning team workload with its actual capabilities, and systematically improving the planning model based on historical data. At the same time, project managers can incorporate additional time or resource contingency buffers into their planning policy for unforeseen events, such as changes in the scope of work, technical issues, or the need to complete urgent tasks. Including such buffers increases the planning process's resilience to uncertainty and helps reduce the risk of delays in the completion of planned work.

In the described case, several types of workloads expressed in man-days were defined: New Unit, Unit Extension, Feature, Large Bug Fix, Small Bug Fix, Technical Improvement. The proposed methodology does not impose a specific classification. Each organization has its own unique types of tasks, and classification helps quantify individual types, as the individual efforts required to deliver them can vary significantly.

Table 2. Example of a Planning Policy

Task type	Average effort [MD]	Average elapsed time [calendar days]	Average elapsed time with contingency buffer [calendar days]
New Unit	30	42	90
Unit Extension	10	21	60
Feature	2	14	30
Large Bug Fix	5	7	14
Small Bug Fix	1	3	7
Technical improvement	3	18	21

Timeline (average elapsed time), expressed in calendar days, is a key parameter for determining the actual duration of individual work items and serves as the basis for resource allocation over time. In software development projects, it is common for work estimated at, for example, 10 Man-Days (MD) to be completed over a period of two months, reflecting the fact that development teams typically work on multiple projects simultaneously. Consequently, effective resource allocation is essential to ensure that the workload assigned in any given month does not exceed the Work Limit, which corresponds to the available Net Capacity. Elapsed time can be derived from historical data stored in project management systems such as Jira and provides the foundation for defining planning policies.

Table 2 presents an example of a Planning Policy derived from historical project data. For each task type, the average development effort expressed in Man-Days and the average elapsed time from task initiation to completion are calculated based on completed tasks stored in work management systems such as Jira. These values constitute the primary input parameters for allocating work within the Planning Matrix and are periodically recalibrated as new historical data becomes available.

Planning Policy is also a tool for communicating the estimated effort and completion time of specific task types to stakeholders. Further research has shown that the most representative value is the average value from the last 12 months, which effectively reduces the impact of deviations. It is also important to remember to add a buffer for contingency for unforeseen events.

Planning Matrix

By understanding Net Capacity, task workloads, and Planning Policy, the project manager can present a plan for task implementation. This is supported by a tool called the Planning Matrix. It's important to note that the project manager can incorporate all steps of the software development process defined within their organization into the plan. The timeline is assumed to be between the first and last task status.

The Planning Matrix is continuously modified by the project manager due to emerging risks and changing task priorities. Additionally, when communicating with stakeholders, the project manager can use the level of certainty for individual process steps, as the more advanced the task is, the more confident the project manager is about the estimated dates.

Table 3 shows the Planning Matrix. The project manager allocates capacity within the defined Net Capacity for a given month, taking into account the task priorities or deadlines expressed by stakeholders. It is important to appropriately distribute resources in accordance with the efforts and the average execution times of individual task categories described in the Planning Policy.

The interpretation of the data presented in the table is as follows: the Key column denotes the task key from a system such as Jira, and

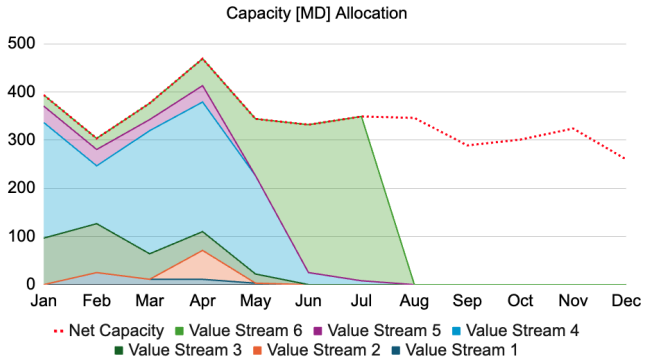

Fig. 2. Ideal visualization of Planning Matrix

the Value stream column indicates that the task builds a specific demand for the organization. The most important field is Task type, as it is used to derive the average effort required to complete a specific development task, calculated based on historical data (Data-Driven Approach). Then, based on the Planning Policy, the elapsed time for that task type is determined.

The Planning Matrix can be visualized as shown in Figure 2. The top dotted line indicates the Net Capacity, which the project manager should not exceed when planning team tasks within a given month due to Work in Progress limits. In reality, overcapacities occur when dates cannot be changed due to deadlines, and work accumulates due to delays.

Individual colors indicate work planned for value streams. Thanks to this, described method allows you to manage not only project plans but also programs that share development resources. The figure represents the ideal situation.

Based on visualization shown in Figure 2, the project manager can estimate the size of the backlog (in terms of the time required to complete it), when development teams have the first available slot to start new tasks, and how much free capacity they still have available in a given month. For example, the chart shows that new tasks can be started at the beginning of August and the project manager has approximately 346 MD at his disposal for this month.

In conclusion of the chapter, it should be noted that the workload required to implement different types of tasks and the related Planning Policy should be calculated based on historical data on a regular basis, which will allow the model to be tuned and react to the changing environment.

4. RESULTS

This chapter describes the results of implementing the described methodology in an organization engaged in large-scale software development. Projects are carried out within programs that encompass value streams. Each program is governed by a roadmap, which is executed through projects with dedicated project plans.

Figure 3 shows a visualization of the real Planning Matrix created during the actual implementation of the described methodology through several months. It's worth mentioning that the approach described is data-driven. This means that the project manager receives ongoing feedback from the system regarding average effort and timelines for specific task types. This allows him to fine-tune the model using the new, updated values. The effect of this approach can be seen in Figure 3, where, at the beginning of the methodology implementation, planned work often exceeded Net Capacity due to underestimated tasks or delays. As the data

Table 3. Planning Matrix

Key	Value stream	Priority	Summary	Task type	Effort [MD]	Team	Dev Start	Dev End	Jan [MD]	Feb [MD]	Mar [MD]	Apr [MD]	May [MD]	Jun [MD]
Key-1	VS 1	5	Implement real-time shipment tracking	New Unit	30	X	Jan-10	Mar-10	10	13	7			
Key-2	VS 2	5	Fix inconsistencies in warehouse inventory	New Unit	30	Y	Feb-05	Apr-05		12	16	2		
Key-3	VS 1	5	Develop automated route optimization	Unit Extension	10	Y	Apr-15	May-15				5	5	
Key-4	VS 1	4	Add validation rules for shipment data	Unit Extension	10	Z	Apr-15	May-15				5	5	
Key-5	VS 5	1	Improve performance of order process	Unit Extension	10	Z	Feb-15	Mar-15		5	5			
Key-6	VS 5	3	Integrate barcode scanning functionality	Unit Extension	10	X	Mar-15	Apr-15			5	5		
Key-7	VS 2	1	Refactor legacy code in the dispatch	Feature	2	X	May-10	May-10					2	
Key-8	VS 1	3	Implement notification system for delay	Feature	2	X	Jun-10	Jun-10						2

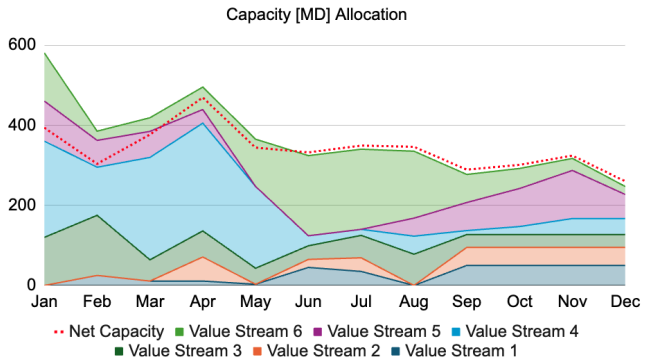

Fig. 3. Visualization of Planning Matrix in real

is refined, it becomes clear that the work was planned with a very high degree of accuracy.

After several planning iterations, the Planning Matrix became the primary operational tool supporting resource allocation across multiple value streams. Historical workload and elapsed time metrics were periodically recalculated using newly completed tasks, allowing the Planning Policy to evolve together with the organization. As a result, planning assumptions became increasingly aligned with the actual execution characteristics of development teams. This iterative calibration process transformed planning from a static estimation exercise into a continuously improving data-driven process.

Another important observation concerns the visibility of resource utilization. The visualization of the Planning Matrix enabled project managers to identify future capacity shortages several months in advance, making it possible to negotiate priorities with stakeholders before over-allocation occurred. Instead of reacting to delays after they appeared, planning activities became proactive, allowing workload balancing across multiple development teams while respecting the defined Net Capacity. This significantly improved the transparency of planning activities and facilitated communication between project managers, product owners, and business stakeholders.

The methodology also improved consistency in planning decisions. Since all workload estimates were derived from the same Planning Policy based on historical project data, individual project managers no longer relied exclusively on subjective expert judgement. This reduced planning variability between different value streams and provided a common reference model for estimating new initiatives. Consequently, delivery expectations communicated to stakeholders became more consistent and easier to justify using empirical evidence.

An additional benefit observed during the implementation was the ability to evaluate the impact of changing priorities. Whenever new initiatives entered the backlog or existing projects changed scope, the Planning Matrix immediately exposed their effect on future capacity allocation and expected delivery dates. This enabled informed decision-making regarding trade-offs between competing business priorities without violating the capacity constraints of development teams.

Finally, the implementation demonstrated that the methodology scales well in environments where multiple projects compete for the same engineering resources. The qualitative outcomes summarized in Table 4 confirm improvements in planning accuracy, capacity utilization, delivery predictability, stakeholder transparency, and overall planning adaptability.

By combining Net Capacity, historical effort estimates, Planning Policy, and the Planning Matrix within a single planning framework, project managers were able to coordinate parallel development activities while maintaining compliance with agile delivery practices. Rather than replacing Agile methodologies, the proposed approach complemented them by introducing a higher-level resource planning mechanism that supported strategic planning across multiple value streams.

5. DISCUSSION

The proposed methodology addresses an important gap between traditional project management and agile software development in environments characterized by multiple value streams, distributed teams, and strong dependencies on external stakeholders. While agile frameworks such as Scrum, Kanban, and their scaled variants provide effective mechanisms for managing development work, they primarily focus on workflow organization and team collaboration rather than long-term resource allocation across competing initiatives. The presented methodology complements these approaches by introducing a structured, data-driven mechanism for planning development capacity without compromising agile delivery principles.

The implementation results demonstrate that planning accuracy improves as historical project data are continuously incorporated into the planning process. This observation confirms one of the key assumptions of the methodology: planning should not rely solely on expert judgment but should evolve through iterative calibration based on empirical evidence. The use of historical workload and elapsed time metrics enables more realistic estimates of development effort and delivery timelines, reducing discrepancies between planned and actual work over time.

Table 4. Qualitative outcomes of the methodology implementation

Observed area	Result of implementation
Planning accuracy	Improved alignment between planned and actual workload through iterative calibration based on historical data.
Capacity utilization	Better utilization of available Net Capacity while reducing over-allocation of development teams.
Resource allocation	More balanced distribution of work across multiple value streams and shared development teams.
Delivery predictability	More realistic delivery forecasts based on empirical effort and elapsed time data.
Stakeholder transparency	Increased visibility of future capacity, priorities, and expected delivery timelines, supporting communication with stakeholders.
Planning adaptability	Faster adjustment of plans in response to changing priorities while preserving overall planning consistency.

An important contribution of the proposed approach is the explicit integration of Net Capacity with a Planning Policy and the Planning Matrix. Rather than treating resource availability as a static constraint, the methodology models organizational factors such as meetings, maintenance activities, training, vacations, and team efficiency to determine the actual development capacity available for value-creating work. This provides project managers with a realistic basis for balancing multiple concurrent initiatives while maintaining transparency for stakeholders.

The Planning Matrix also offers practical benefits beyond resource allocation. By combining task workload, historical delivery timelines, and capacity constraints within a single planning model, it enables project managers to estimate backlog duration, identify the first available implementation slots, evaluate the impact of priority changes, and communicate delivery expectations with greater confidence. In organizations where development teams simultaneously support multiple business programs, this visibility represents a significant operational advantage over traditional backlog-based planning approaches.

Although the methodology demonstrated promising results in a real industrial environment, several limitations should be acknowledged. The proposed planning model was developed and validated within a single organization operating a software platform with specific characteristics, including multiple customer value streams and dependencies on external partners. Consequently, some planning parameters, task classifications, and capacity models may require adaptation before being applied in different organizational contexts. Furthermore, the quality of planning outcomes depends directly on the availability and consistency of historical project data, making reliable data collection an essential prerequisite for successful adoption.

The methodology also creates opportunities for further automation. Since planning decisions are derived from structured planning rules combined with empirical project data, many planning activities could be supported by intelligent software agents capable of continuously updating planning parameters, detecting capacity conflicts, recommending alternative scheduling scenarios, and assisting project managers in decision-making. Such developments represent a natural continuation of the proposed methodology and may further improve planning efficiency in large-scale software development environments.

Future research

A promising direction for future research is the development of an intelligent planning agent that automates the proposed methodology while preserving the role of the project manager as the final decision-maker. The agent would utilize historical project data, planning policies, Net Capacity calculations, and the Planning Matrix to generate and continuously update resource allocation plans. Rather than replacing human expertise, the agent would support planning decisions by recommending task schedules, identifying capacity conflicts, estimating delivery dates, and proposing alternative allocation scenarios based on organizational priorities.

The proposed agent would combine deterministic planning rules with data-driven reasoning. Historical data collected from work management systems such as Jira would be used to calibrate workload estimates and task completion timelines, while the Planning Policy would define organizational constraints and resource allocation rules. This hybrid approach would enable adaptive planning in dynamically changing environments where priorities, dependencies, and resource availability evolve continuously.

Recent advances in artificial intelligence, particularly in large language models (LLMs) and autonomous AI agents, demonstrate their increasing potential to support complex decision-making and software engineering activities. Recent surveys indicate that LLM-based agents are evolving from conversational assistants toward autonomous systems capable of planning, reasoning, interacting with external tools, and solving complex multi-step tasks [16, 17]). These capabilities make AI agents a promising technology for supporting resource planning and decision-making in large-scale software development environments.

The research presented in this paper serves as the methodological foundation for the development of such an intelligent planning agent. Building upon the proposed Planning Matrix methodology, implementation work is currently at an advanced stage. A proof-of-concept agent has already been developed, and preliminary experimental results demonstrate that the methodology can be successfully operationalized to automatically generate resource allocation plans from historical project data, Planning Policies, and Net Capacity constraints. The initial implementation confirms the feasibility of combining deterministic planning algorithms with AI-assisted reasoning to support project managers while maintaining human supervision over final planning decisions.

Future work will investigate the integration of large language models with structured planning algorithms to enable natural language interaction with project managers. The long-term objective is to develop an intelligent decision-support system that combines explainable AI with empirical planning models, providing practical support for resource planning in large-scale software development environments.

6. CONCLUSIONS

The study presented a data-driven methodology for planning software development resources in complex environments characterized by multiple value streams, distributed teams, and dependencies on external stakeholders. The proposed approach effectively bridges the gap between traditional project management

and agile product development, enabling organizations to preserve agility while achieving greater predictability and control over resource allocation.

The methodology integrates several key elements into a coherent framework. The concept of net capacity allows for a realistic estimation of the actual effort available for development work by incorporating organizational constraints and human factors. The data-driven approach, based on historical data from systems such as Jira, enables continuous refinement of effort estimation and delivery timelines. The planning policy formalizes resource allocation rules and incorporates buffers for uncertainty, while the planning matrix provides a practical tool for visualizing and coordinating work across multiple value streams within defined capacity limits.

The implementation results indicate that the methodology significantly improves planning accuracy over time. Initial discrepancies between planned and actual workloads, primarily caused by underestimation and delays, are gradually reduced through iterative calibration based on empirical data. As a result, the alignment between planned and executed work increases, confirming the effectiveness of the proposed approach in real-world conditions.

From a practical standpoint, the methodology supports more efficient utilization of development team capacity, enhances transparency for stakeholders, and improves the ability to manage parallel projects and competing priorities. It also contributes to reducing inefficiencies associated with planning in large-scale and dynamically changing environments.

However, the approach is not without limitations. It was developed and validated within a specific organizational context, which may limit its direct applicability to other settings. Its effectiveness also depends on the availability and quality of historical data, as well as the organization's maturity in data collection and process standardization.

Future research may focus on validating the methodology across different industries and organizational scales, integrating predictive analytics or machine learning techniques to further improve estimation accuracy, extending the model to include financial and business value dimensions, and exploring opportunities for automation within planning tools.

In conclusion, the proposed methodology provides a pragmatic and scalable solution for resource planning in large-scale software development. By combining empirical data with structured planning mechanisms, it enables organizations to better manage complexity while maintaining the core principles of agile software development.

7. REFERENCES

[1] Adam Alami, Mansooreh Zahedi, and Oliver Krancher. The role of psychological safety in promoting software quality in agile teams. *Empirical Software Engineering*, 29:119, 2024.

[2] Filipe Almeida, José Simões, and Sandra Lopes. Large-scale agile frameworks: A comparative review. *Journal of Systems and Software*, 173:110852, 2021.

[3] E. C. Daraojimba, C. N. Nwasike, A. O. Adegbite, C. A. Ezeigweneme, and J. O. Gidiagba. Comprehensive review of agile methodologies in project management. *Computer Science & IT Research Journal*, 5(1):190–218, 2024.

[4] Helen Edison, Xiaofeng Wang, Pekka Abrahamsson, and Kieran Conboy. Large-scale agile software development: A systematic review and research agenda. *IEEE Transactions on Software Engineering*, 48(8):2731–2753, 2022.

[5] R. D. Estrada-Esponda, M. Lopez-Benítez, G. Matturro, and J. C. Osorio-Gomez. Selection of software agile practices using analytic hierarchy process. *Heliyon*, 10:e22948, 2024.

[6] Ahmed Fawzy, Amjed Tahir, Matthias Galster, and Peng Liang. Exploring data management challenges and solutions in agile software development: A literature review and practitioner survey. *Empirical Software Engineering*, 30:77, 2025.

[7] Nicole Forsgren, Jez Humble, and Gene Kim. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press, 2018.

[8] Annabelle Gawer. Digital platforms and ecosystems: Remarks on the dominant organizational forms of the digital age. *Journal of Business Research*, 144:124–135, 2022.

[9] Mariam Gómez Sánchez, Eduardo Lalla-Ruiz, Alejandro Fernández Gil, Carlos Castro, and Stefan Voß. Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research*, 309(3):958–976, 2023.

[10] A. Hossain. A systematic mapping study on scrum and kanban in software development. Master's thesis, University of Turku, 2023.

[11] L. Mayo-Alvarez, S. Del-Aguila-Arcentales, A. Alvarez-Risco, M. C. Sekar, N. M. Davies, and J. A. Yáñez. Innovation by integration of drum-buffer-rope (dbr) method with scrum-kanban and use of monte carlo simulation for maximizing throughput in agile project management. *Journal of Open Innovation: Technology, Market, and Complexity*, 10:100228, 2024.

[12] Mary Poppendieck and Tom Poppendieck. *Lean Thinking for Software Development*. Addison-Wesley, 2013.

[13] M. Poženel, L. Fürst, D. Vavpotič, and T. Hovelja. Agile effort estimation: Comparing the accuracy and efficiency of planning poker, bucket system, and affinity estimation methods. *International Journal of Software Engineering and Knowledge Engineering*, 2023.

[14] Abhijit Putta, Maria Paasivaara, and Casper Lassenius. Benefits and challenges of adopting the scaled agile framework (safe): Preliminary results from a multivocal literature review. In *Product-Focused Software Process Improvement*, pages 334–351. Springer, 2018.

[15] M. Shafique and S. I. Sohail. Pros and cons of implementing agile methodologies on large-scale projects. *International Journal of Contemporary Issues in Social Sciences*, 3(1):493–510, 2024.

[16] L. Wang, C. Ma, W. Feng, Z. Zhang, H. Liu, Y. Zhao, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[17] Z. Xi, W. Chen, X. Guo, W. Wang, Y. Ge, X. Zhang, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint*, arXiv:2309.07864, 2023.

[18] N. Yevtushenko and T. Makarov. Kanban and scrum methods in consulting: Advantages and disadvantages. In *Theoretical and Empirical Scientific Research: Concept and Trends*. Logos Science, 2024.