UDC 004.8:159.955:005.32
DOI https://doi.org/10.15589/znp2026.1(503).2.11

MODELING THE BALANCE BETWEEN COGNITIVE FOCUS AND TEAM EFFECTIVENESS IN INTELLIGENT NOTIFICATION FILTERING SYSTEMS WITHIN AGILE TEAMS

МОДЕЛЮВАННЯ БАЛАНСУ КОГНІТИВНОЇ КОНЦЕНТРАЦІЇ ТА КОМАНДНОЇ ЕФЕКТИВНОСТІ В СИСТЕМАХ ІНТЕЛЕКТУАЛЬНОЇ ФІЛЬТРАЦІЇ СПОВІЩЕНЬ В AGILE-КОМАНДАХ

Vadym I. Ziuziun
vadym.ziuziun@knu.ua
ORCID: 0000-0001-6566-8798

Anastasiia O. Avramets
nastia.avramec11.01@gmail.com
ORCID: 0009-0009-6062-1144

Anna S. Kolomiiets
anna.kolomiiets@knu.ua
ORCID: 0000-0003-4252-5975

Ihor V. Miroshnychenko
ihor.miroshnychenko@knu.ua
ORCID: 0000-0002-1307-7889

В. І. Зюзюн,
канд. техн. наук, доцент

А. О. Аврамець,
здобувачка вищої освіти

А. С. Коломієць,
канд. екон. наук, доцент

І. В. Мірошниченко,
канд. екон. наук, доцент

Taras Shevchenko National University of Kyiv, Kyiv
Київський національний університет імені Тараса Шевченка, м. Київ

Abstract. *The purpose* of the study is to develop a mathematical model for adaptive notification filtering that makes it possible to optimize the balance between an individual developer's cognitive focus and collective efficiency (Velocity) within a dynamic Agile environment.
Method. The research methodology is based on an integrated combination of graph theory to model dynamic logical dependencies among tasks performed by members of an Agile team, and queueing theory to analyze incoming notification streams. To formalize individual cognitive losses, the study employs mathematical modeling of nonlinear exponential relationships that describe the process of concentration recovery («cognitive viscosity») as a function of uninterrupted work duration. Team-level losses are assessed by computing the expected value of idle time through stochastic analysis of specialists' backlog states and by determining node centrality metrics within the project dependency graph. Optimization of notification delivery timing is achieved by solving a global objective maximization problem using adaptive weighting coefficients that vary in accordance with the project phase.
Results. The study resulted is a mathematical model and an intelligent notification-filtering algorithm that enable quantitative balancing between an individual developer's productivity and overall team velocity. The research formalizes an autonomy buffer criterion that allows the system to anticipate critical moments when colleagues' parallel tasks are nearing exhaustion and to proactively prevent blockage of the project's critical path. It is demonstrated that the use of a dynamic sprint-progress coefficient ensures a flexible transition from an aggressive cognitive-focus protection strategy at the beginning of an iteration to the prioritization of immediate communication during the release phase. Situational analysis confirms that the proposed approach makes it possible to transform the subjective importance of messages into measurable economic and temporal metrics (Cost of Delay, Lead Time), minimizing losses caused by interruptions to the flow state without compromising team synchronization.
The scientific novelty of the study lies in the development of a comprehensive mathematical model for adaptive notification filtering that, for the first time, integrates the «cognitive viscosity» of an individual developer with a dynamic graph of team dependencies within the Agile cycle. Unlike existing approaches that treat the specialist as an isolated unit, the proposed model formalizes the nonlinear relationship between concentration recovery time and the depth of immersion in the «flow state», and introduces an adaptive sprint-progress coefficient to achieve Pareto-efficient balancing between the protection of individual cognitive resources and the minimization of team-level losses. This makes it possible to quantitatively justify the transition from static filtering regimes to context-aware

communication management, in which message delivery priorities dynamically adjust according to the project phase and the probability of blocking the critical path.
The practical significance of the obtained results lies in establishing an algorithmic and methodological foundation for the development of intelligent notification filtering systems capable of dynamically adapting to the phases of an IT project's life cycle. Implementing the proposed models in corporate messengers and project management systems will make it possible to automate the calculation of an «autonomy buffer» for each developer and to apply an intelligent message «batching» mechanism.
This will enable the minimization of team losses (Cost of Delay) and the optimization of overall development speed (Lead Time) by eliminating cascading idle time caused by communication delays along critical paths. The proposed approach makes it possible to transform the subjective importance of information flows into a measurable quantity, ensuring a balance between protecting an individual's «flow state» and achieving the collective goals of an Agile team.
Key words: information technology; intelligent notification filtering; mathematical modeling; cognitive processes; Agile; Agile team; team velocity; communication processes; cognitive viscosity; flow state.

Анотація. *Метою дослідження* є розробка математичної моделі адаптивної фільтрації сповіщень, яка дозволяє оптимізувати баланс між індивідуальною когнітивною концентрацією розробника та колективною ефективністю (Velocity) в умовах динамічного Agile-середовища.
Методика. Методика дослідження базується на комплексному поєднанні апарату теорії графів для моделювання динамічних логічних залежностей між завданнями учасників Agile-команди та теорії масового обслуговування для аналізу вхідних потоків сповіщень. Для формалізації індивідуальних когнітивних втрат використано математичне моделювання нелінійних експоненціальних залежностей, що описують процес відновлення концентрації («когнітивну в'язкість») залежно від тривалості неперервної роботи. Оцінка командних втрат здійснюється шляхом обчислення математичного сподівання простоїв на основі стохастичного аналізу стану беклогу фахівців та визначення показників центральності вузлів у графі проєктних залежностей. Оптимізація моменту доставки повідомлень реалізована через розв'язання задачі максимізації глобальної цільової функції з використанням адаптивних вагових коефіцієнтів, що змінюються відповідно до фази.
Результати. Результатом дослідження є розроблена математична модель та алгоритм інтелектуальної фільтрації сповіщень, які дозволяють кількісно збалансувати індивідуальну продуктивність розробника та колективну швидкість команди (Velocity). У ході роботи формалізовано критерій «запасу автономності», який дозволяє системі прогнозувати критичні моменти вичерпання паралельних задач у колег і запобігати блокуванню критичного шляху проєкту. Доведено, що використання динамічного коефіцієнта прогресу спринту забезпечує гнучкий перехід від стратегії агресивного захисту когнітивного фокусу на початку ітерації до пріоритезації миттєвої комунікації на етапі релізу. Ситуаційний аналіз підтвердив, що запропонований підхід дозволяє трансформувати суб'єктивну важливість повідомлень у вимірювані економічні та часові метрики (Cost of Delay, Lead Time), мінімізуючи втрати від переривання «стану потоку» без шкоди для командної синхронізації.
Наукова новизна дослідження полягає у розробці комплексної математичної моделі адаптивної фільтрації сповіщень, яка вперше інтегрує функцію «когнітивної в'язкості» індивідуального розробника з динамічним графом командних залежностей в межах Agile-циклу. На відміну від існуючих підходів, що розглядають фахівця як ізольовану одиницю, запропонована модель формалізує нелінійну залежність часу відновлення концентрації від глибини занурення у «стан потоку» та вводить адаптивний коефіцієнт прогресу спринту для Парето-ефективного балансування між захистом індивідуального когнітивного ресурсу та мінімізацією командних втрат. Це дозволило кількісно обґрунтувати перехід від статичних режимів фільтрації до контекстно-залежного управління комунікаціями, де пріоритетність доставки повідомлень динамічно змінюється залежно від фази проєкту та імовірності блокування критичного шляху.
Практичне значення отриманих результатів полягає у створенні алгоритмічного та методичного підґрунтя для розробки інтелектуальних систем фільтрації сповіщень, здатних динамічно адаптуватися до фаз життєвого циклу ІТ-проєкту. Впровадження розроблених моделей у корпоративні месенджери та системи управління проєктами дозволить автоматизувати розрахунок «запасу автономності» для кожного розробника та застосовувати механізм інтелектуального «пакетування» повідомлень.
Це забезпечить можливість мінімізувати командні втрати (Cost of Delay) та оптимізувати загальну швидкість розробки (Lead Time) шляхом усунення каскадних простоїв через затримки комунікації на критичних шляхах. Запропонований підхід дозволить трансформувати суб'єктивну важливість інформаційних потоків у вимірювану величину, забезпечуючи баланс між захистом індивідуального «стану потоку» та досягненням колективних цілей Agile-команди.
Ключові слова: інформаційні технології; інтелектуальна фільтрація сповіщень; математичне моделювання; когнітивні процеси; Agile; Agile-команда; командна ефективність; комунікаційні процеси; когнітивна в'язкість; стан потоку.

FORMULATION OF THE PROBLEM

In modern software engineering, a specialist's productivity is directly linked to their ability to maintain prolonged and uninterrupted cognitive focus, often referred to in psychology as the «flow state». In this state, a developer can manipulate complex mental models of code and architecture with maximum efficiency. Within the context of software engineering, this psychological phenomenon directly impacts the quality of architectural decisions, the speed of feature implementation, and the number of errors arising during the development of complex software systems.

However, contemporary Agile methodologies (such as Scrum or Kanban) are built on principles of high-intensity communication, which is necessary to ensure continuity in development processes and rapid adaptation to changes. This creates a fundamental contradiction between individual and team productivity:

A. *Cognitive aspect*. Individual attention-protection mechanisms, such as filtering or ignoring notifications, are necessary to maintain focus. However, any delay in responding to incoming information creates communication gaps.

B. *Systemic aspect*. In the context of tightly interdependent tasks within an IT team, a delayed response from a single specialist can lead to cascading idle time for other project participants. This is critical for tasks on the development's critical path, where a delay in one component blocks the work of the entire team.

C. *Limitations of existing solutions*. Traditional notification management systems (DND modes, static filters) typically operate under a limited set of rules that do not account for the Cost of Delay for the team at a given moment. They treat the user as an isolated unit, ignoring the dynamics of the sprint and the specialist's current role within the team dependency graph.

Thus, the task of formalizing a quantitative relationship between a developer's individual cognitive losses and team efficiency metrics (such as Velocity and Lead Time) within the Agile cycle becomes highly relevant, necessitating the development of adaptive mathematical models for intelligent notification filtering.

ANALYSIS OF RECENT RESEARCH AND PUBLICATIONS

Research in cognitive psychology, including studies conducted in IT environments [1–3], confirms that context recovery after an interruption takes between 10 and 15 minutes, leading to the degradation of mental task models. Study [4] examined the impact of incoming notifications on work process quality.

Existing technical solutions, such as IBM's Intelligent Notification System [5] or the PrefMiner system [6, 7], leverage contextual data (e.g., calendar, location) and machine learning to predict the importance of messages. These systems represent the current direction of research in intelligent notification filtering, focusing on the use of contextual information and machine learning methods to assess message relevance. However, most existing models treat the specialist as an isolated unit.

Research [8] highlights the need to account for the dynamics of Agile teams, as well as the application of intelligent technologies in IT project management under the Scrum framework [9, 10]. Yet, previous studies have not proposed a formalized mathematical framework that links individual cognitive resources to group development velocity (Velocity).

In [11], V. Ziuziun and A. Avramets developed a conceptual model of an intelligent notification filtering system, integrating user context analysis, dynamic profiling, and strict service-level constraints (SLA) for critical events. This work laid the foundation for the research presented in this publication.

Since existing approaches largely ignore the team development context, do not account for the dynamic structure of task dependencies, and do not allow for the assessment of team losses caused by communication delays within a sprint or iteration, the investigation of this problem remains highly relevant.

SEPARATION OF PREVIOUSLY UNRESOLVED PARTS OF THE OVERALL PROBLEM

The question of quantitatively measuring the negative impact of individual notification filtering on team metrics (Lead Time, Cycle Time) remains unresolved. In particular, the following gaps can be highlighted:

– the absence of mathematical relationships linking notification delays to the likelihood of blocking the project's critical path;

– the lack of cognitive viscosity models that account for the nonlinear nature of concentration recovery depending on the depth of prior immersion in work;

– the absence of algorithms for dynamically allocating priorities based on the phase of the sprint lifecycle.

PURPOSE OF THE RESEARCH

The purpose of this study is to develop a mathematical model for adaptive notification filtering that optimizes the balance between a developer's individual cognitive concentration and collective efficiency (Velocity) within a dynamic Agile environment.

METHODS, OBJECT AND SUBJECT OF RESEARCH

Research object. The processes of information interaction and resource coordination in distributed IT teams using Agile methodologies.

Research subject. Mathematical models and algorithms for intelligent notification filtering that account for the dynamics of a developer's cognitive state and the structure of team dependencies.

Research methods. Graph theory (modeling task dependencies), queuing theory (analysis of notification flows), multi-objective optimization, and stochastic modeling (Monte Carlo simulation).

PRESENTATION OF THE MAIN MATERIAL

In modern software engineering, a specialist's productivity directly depends on their ability to maintain prolonged and uninterrupted cognitive focus, often referred to as the «flow state» [12]. In this state, a developer manipulates complex mental models of code and architecture with maximum efficiency, which directly affects the quality of decisions and the speed of feature implementation. However, the nature of Agile methodologies, such as Scrum or Kanban, requires high-intensity communication to enable rapid adaptation to changes, creating a fundamental conflict between individual focus and team efficiency. Any delay in responding to incoming information can lead to cascading idle time and blockages along critical development paths [13], while immediate interruptions degrade concentration, the recovery from which requires substantial time.

Existing notification management systems typically operate under static rules and treat the user as an isolated unit, ignoring sprint dynamics and the developer's current role within the team dependency graph. There is a pressing need for intelligent models capable of assessing the Cost of Delay for the entire team and comparing it with an individual developer's cognitive losses. Such an approach allows not merely blocking notifications but adaptively managing delivery priorities depending on the project lifecycle phase – from aggressively protecting focus at the beginning of the sprint to immediate synchronization before the release.

To implement such balancing, it is necessary to move beyond qualitative descriptions and formalize a quantitative relationship between an individual's psychophysiological state and the efficiency metrics of the entire Agile team. This requires the development of a mathematical framework that integrates a developer's «cognitive viscosity» function with a dynamic analysis of the team dependency graph.

Let us now examine in more detail the mathematical modeling of team losses that arise from delayed notifications.

1. Mathematical modeling of team losses

We represent the team as a directed graph [13–15]:

$$G = (V, E), \quad (1)$$

where V represents the specialists, and E represents the logical dependencies between their tasks. For each notification s delayed by t_{del}, there exists a mathematical expectation of team losses E_{loss} [12]. Team losses occur when a notification is delayed beyond a colleague's autonomy reserve of tasks.

To quantitatively assess this effect, we introduce the expected team loss, defined as:

$$E_{loss}(s, t_{del}) = \sum_{j=1}^{M} \left(1 - e^{-\mu_j \cdot t_{del}}\right) \cdot C_j \cdot \left(t_{del} - t_{alt,j}\right)^{+}, \quad (2)$$

where μ_j s the intensity parameter for the exhaustion of a colleague's alternative tasks. To improve the accuracy of the model, the task-exhaustion intensity parameter is proposed to be calculated based on the current state of the specialist's backlog in the project management system [12]:

$$\mu_j = \frac{1}{\sum_{p=1}^{n} w_p \cdot t_{est,p}}, \quad (3)$$

where $t_{est,p}$ – the estimated completion time of specialist j $p-th$ active task;

w_p – the priority coefficient of the task. This approach accounts for the fact that the risk of idle time (Cost of Delay) increases exponentially as the number of available alternative tasks for colleagues decreases;

$\left(1 - e^{-\mu_j \cdot t_{del}}\right)$ – an exponential factor formalizing the probability that a colleague's alternative tasks will be exhausted. The longer the delay t_{del}, the closer the value approaches 1, representing complete blockage;

C_j – the cost per hour of specialist j. his parameter introduces an economic dimension, allowing E_{loss} to be measured in monetary or resource equivalents;

$t_{alt,j}$ – he time during which a colleague can remain productive without receiving a response (availability of parallel tasks);

$\left(t_{del} - t_{alt,j}\right)^{+} = \max\left(0, t_{del} - t_{alt,j}\right)$ – an operator formalizing that team losses begin to accrue only after the internal autonomy reserve of specialist j is exhausted. Mathematically, this introduces the concept of a «developer's autonomy reserve»: losses start accumulating only when the waiting time exceeds the time the colleague can spend on parallel tasks.

2. Cognitive viscosity function

Cognitive viscosity refers to a property of a developer's mental state that determines the complexity and duration of recovering the work context after an interruption.

To describe individual developer losses, we use the concentration recovery function τ_{rec}, which accounts for the nonlinear nature of accumulated cognitive load:

$$\tau_{rec}(T_{work}) = \tau_{min} + \sigma \cdot \left(1 - e^{-k \cdot T_{work}}\right), \quad (4)$$

where T_{work} – the duration of uninterrupted deep work before receiving a notification;

τ_{min} – the minimum time required to switch attention;

σ – the *cognitive viscosity coefficient*, representing the maximum theoretical context recovery time for a specific developer;

k – the immersion rate parameter, reflecting the complexity of the current mental task model.

The use of the exponential function $\sigma \cdot \left(1 - e^{-k \cdot T_{work}}\right)$ appropriately captures the psychological effect of « depth of immersion».

This formula mathematically confirms that interruptions after a prolonged work cycle $(T_{work} > 90 min)$ cause significantly greater harm than concentration degradation at the beginning of the cycle. For small T_{work} (start of work), losses are minimal ($\approx \tau_{min}$), whereas during extended work sessions $(T_{work} > 90 \text{min})$ hey asymptotically approach the maximum ($\tau_{min} + \sigma$). This supports the assertion that interruptions during the "flow" state are the most destructive.

3. Global optimization objective function

The core of the system's decision-making is to solve the problem of maximizing collective efficiency F, interpreted as a generalized utility [16-17], which accounts for both the individual cognitive resources and team losses due to communication delays:

$$\max F = \sum_{i=1}^{N} \left(\alpha \cdot T_{focus,i} - \beta \cdot \tau_{rec,i} \right) - \gamma(R) \cdot \sum_{s \in S} E_{loss}(s, t_{del}), \quad (5)$$

where $T_{focus,i}$ is the predicted time that specialist i will remain in a state of concentration;

α, β – weighting coefficients for prioritizing deep work and minimizing fatigue, respectively;

$\gamma(R)$ – dynamic sprint progress coefficient, calculated as:

$$\gamma(R) = \gamma_0 \cdot e^{R^2}, \quad (6)$$

$$R = \frac{t_{current}}{t_{sprint\_end}}. \quad (7)$$

This provides flexibility: at the beginning of the sprint ($R \to 0$), the system aggressively protects developers' focus, whereas prior to the release ($R \to 1$), the priority shifts to immediate team communication.

4. Algorithmic implementation

The algorithmic implementation of the proposed model is based on a combination of graph theory methods, stochastic optimization, and analysis of the user's cognitive context. The intelligent notification filtering system treats each incoming message as an event that can potentially shift the balance between a developer's individual cognitive concentration and the collective efficiency of the Agile team.

The algorithm is based on a dynamic, directed, weighted graph of team dependencies $G_t = (V, E)$, where the set of vertices V represents team members (or their active tasks), and the set of edges E reflects the current logical and temporal dependencies between their tasks within the sprint. The graph is time-varying and is updated whenever task statuses change in the project management systems.

The decision-making algorithm for message delivery consists of several interconnected stages.

A. *Semantic analysis of the message*. The vector representation of a message is compared with vectorized representations of active tasks, code fragments, or documentation that the developer is working on. The result is a relevance coefficient $R \in [0,1]$, я which is used as a weighting factor in subsequent optimization.

B. *Analysis of the team dependency graph*. In the second stage, the position of the message recipient within the graph G is analyzed using classical graph algorithms.

First, the critical path algorithm is applied to determine whether the developer's current task lies on the sprint's critical path. An interruption of a specialist on the critical path is considered a factor that significantly increases the risk of team losses.

Secondly, for each node in the graph, centrality metrics – particularly *betweenness centrality* – are calculated to quantitatively characterize a specialist's role in transmitting informational and logical dependencies between other team members. A high centrality value is interpreted as an increased team sensitivity to communication delays with that participant.

Additionally, breadth-first search (BFS) algorithms with time constraints are used to assess the reachability of nodes where delayed responses could trigger cascading idle time within a predicted time window. This approach enables the detection of hidden dependencies that are not directly on the critical path but can still cause disproportionate team losses.

C. *Assessment of team and individual losses*. In the third stage, a quantitative evaluation of alternative message delivery scenarios is performed [18]. For each permissible delivery time t, the following are calculated:

– individual cognitive losses of the developer, based on the cognitive viscosity function;

– expected team losses, which depend on the structure of the dependency graph, colleagues' autonomy reserves, and the centrality measures of the recipient.

Taking into account the current sprint phase, a dynamic progress coefficient is introduced, which adaptively adjusts the balance between protecting individual focus and minimizing team delays. Thus, in the early stages of the sprint, the system prioritizes deep concentration, whereas in the final stages, the emphasis shifts toward immediate communication.

D. *Optimization of delivery timing*. The final decision is made by maximizing a global utility function. For each delivery scenario, a value of generalized efficiency is determined, after which the delivery time t, is selected to achieve the optimal balance between individual and team productivity levels.

If the optimal delivery time falls within the allowable delay threshold, the message is delivered immediately. Otherwise, the system applies a notification bundling mechanism, aligning delivery with predicted natural pauses in the developer's workflow.

5. Computational complexity and practical feasibility.

The proposed algorithm employs graph-based procedures with polynomial complexity. The main operations (BFS, centrality calculations, critical path analysis) can be performed incrementally as task states change, ensuring the approach is scalable for medium– and large-sized teams. Practical integration of the algorithm is feasible as a middleware service between corporate messaging platforms and project management systems.

Table 1 presents an example of a situational case, illustrating how the notification management tools proposed in this study can be applied in Agile teams.

The analysis of the example presented in Table 1 demonstrates the practical implementation of the developed mathematical model through a comparison of two polar states of the Agile cycle. The scenario

Table 1. Example of a situational case

Scenario: Development of the «Payment Gateway» feature	
Backend Developer (Team member 1) → Working on a complex encryption algorithm. ***Frontend Developer (Team member 2)*** → Waiting for the updated API documentation to complete the integration of the payment form.	
Situation A. Beginning of the Sprint.	***Situation B: Two days before release.***
Context. Team member 1 has been in a state of concentration for 100 minutes (T_{work} = 100).	***Context. Team member 1*** has been in a state of concentration for 100 minutes (T_{work} = 100).
Event. Team member 2 sends a message: «Can you clarify the JSON format for the expiry_date field?».	***Event. Team member 2*** sends a message: «Can you clarify the JSON format for the expiry_date field?».
System calculation: – Since it is the beginning of the sprint ($R \to 0$), the dynamic coefficient $\gamma(R)$ has a low value. – The cognitive viscosity function τ_{rec} indicates that an interruption now would cost ***Team member 1*** approximately 20 minutes to restore context. – ***Team member 2*** has an «autonomy reserve» (t_{alt}). She has another frontend task in the backlog that she can work on for the next 2 hours.	System calculation: – Now $R \to 1$,(end of the sprint), so the coefficient $\gamma(R)$ grows exponentially. – Analysis of the dependency graph shows that ***Team member 2's*** task is now on the critical path – without this response, the feature release is blocked. – ***Team member 2's*** autonomy reserve is depleted ($t_{alt} = 0$), and any waiting time generates a high *Cost of Delay* (E_{loss}) for the entire team.
Decision. The system *bundles the notification*. ***Team member 2*** does not receive an immediate response, while ***Team member 1*** continues working without losing focus.	***Decision.*** The system delivers the notification *immediately*, despite ***Team member 1*** being in a flow state. The team goal (release) takes priority over individual comfort.

describes a typical development dependency: a frontend specialist cannot complete the integration of a payment form without the API specification being prepared by a backend developer.

In *Situation A* (the beginning of the sprint), the system prioritizes individual cognitive concentration. Since the sprint progress coefficient R is close to zero and a colleague has an «autonomy reserve» in the form of alternative tasks, the system bundles notifications, preventing a 20-minute loss associated with restoring the flow state.

By contrast, in *Situation B* (the end of the sprint), priorities shift: a high value of $\gamma(R)$ and the absence of autonomous resources on the colleague's side make any delay critical for the release. In this case, the system opts for immediate delivery despite the recipient's high cognitive viscosity, because team losses (E_{loss}) outweigh the benefits of individual focus.

The presented situational analysis confirms that the proposed model makes it possible to transform the subjective notion of message «importance» into a measurable quantity grounded in objective project metrics. The use of adaptive weighting coefficients ensures system flexibility, enabling it to autonomously balance the protection of an individual developer's intellectual capital with the team's overall velocity.

This approach addresses the key limitation of existing static filters – their inability to respond to changing context and moment-specific criticality within an iteration. Thus, the algorithm provides dynamic resource optimization in which individual comfort yields to the collective goal only when justified by the risk of blocking the entire development process.

DISCUSSION OF THE OBTAINED RESULTS

The obtained results confirm the hypothesis that the effectiveness of intelligent notification filtering in IT teams cannot be based solely on the analysis of an individual user's state. The key advantage of the developed model lies in the transition from static Do Not Disturb (DND) modes to dynamic balancing, in which each interruption is evaluated through the lens of the project-wide Cost of Delay.

The mathematical formalization of «cognitive viscosity» makes it possible to objectively assess the damage caused by interruptions to the flow state. In particular, the model shows that during prolonged work sessions ($T_{work} > 90\,min$) the concentration recovery time τ_{rec} asymptotically approaches its maximum value, making interruptions in this period the most destructive.

An important aspect of the discussion is the introduction of the dynamic sprint progress coefficient $\gamma(R)$. The situational analysis (Table 1) demonstrated that the system is capable of contextual adaptation. At the beginning of a sprint, priority is given to preserving individual «intellectual capital», as the presence of an «autonomy reserve» among colleagues mitigates the risk of idle time.

Prior to release, the model automatically shifts its focus toward collective velocity, since the probability of blocking the critical path increases, and any delay in communication leads to cascading losses.

The proposed algorithm resolves a fundamental contradiction between individual concentration and team interaction in Agile – an issue that has been largely ignored by most existing solutions, which typically treat a specialist as an isolated unit.

CONCLUSIONS

The study presents a comprehensive investigation of the problem of managing information flows in Agile teams and proposes new approaches to the intelligent filtering of notifications.

1. A comprehensive mathematical model has been developed that, for the first time, integrates the function of an individual developer's «cognitive viscosity» with a dynamic graph of team dependencies. This makes it possible to quantitatively measure the balance between individual focus and collective development velocity.

2. A formalization of team losses (E_{loss}) based on graph theory and backlog analysis is proposed. The introduction of the «autonomy reserve» operator enables the system to predict the moment when a delayed response will begin to genuinely block the work of other sprint participants.

3. An algorithm for the dynamic allocation of priorities has been formulated, based on the semantic analysis of messages and the computation of centrality measures within the dependency graph. The use of an adaptive coefficient $\gamma(R)$ ensures flexible control depending on the phase of the project life cycle.

4. The practical significance of the model has been demonstrated through situational analysis of development scenarios. The implementation of such algorithms makes it possible to minimize the Cost of Delay and optimize Lead Time by intelligently «bundling» messages during predicted pauses in specialists' work.

The developed models provide a scientific and methodological foundation for the creation of modern intelligent development support systems capable of autonomously balancing the protection of human cognitive resources with the achievement of the shared goals of an Agile team.

REFERENCES

[1] Parnin, C., Gorg, C., & Rugaber, S. (2009). TaskBoard: Tracking pertinent task artifacts and plans. 2009 IEEE 17th International Conference on Program Comprehension, Vancouver, BC, Canada, pp. 317-318, https://doi.org/10.1109/ICPC.2009.5090076

[2] Parnin, C., Görg, C., & Rugaber, S. (2010). CodePad: interactive spaces for maintaining concentration in programming environments. In Proceedings of the 5th international symposium on Software visualization (SOFTVIS '10). Association for Computing Machinery, New York, NY, USA, pp. 15–24. https://doi.org/10.1145/1879211.1879217

[3] Parnin, C., & Rugaber, S. (2012). Programmer information needs after memory failure. 2012 20th IEEE International Conference on Program Comprehension (ICPC), Passau, Germany, pp. 123-132, https://doi.org/10.1109/ICPC.2012.6240479.

[4] Mehrotra, A., & Musolesi, M. (2018). Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges. Computer Science. Human-Computer Interaction. https://doi.org/10.48550/arXiv.1711.10171

[5] Bazinette, V., Cohen, N. H., & Ebling M. R., et al. (2001). An Intelligent Notification System. IBM Research Report. Retrieved from: https://www.researchgate.net/publication/228970929_An_intelligent_notification_system

[6] Mehrotra, A., Hendley, R., & Musolesi, M. (2016). PrefMiner: mining user's preferences for intelligent mobile notification management. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16). Association for Computing Machinery, New York, NY, USA, pp. 1223–1234. https://doi.org/10.1145/2971648.2971747

[7] Mehrotra, A., Hendley, R., & Musolesi, M. (2017). Interpretable Machine Learning for Mobile Notification Management: An Overview of PrefMiner. GetMobile: Mobile Comp. and Comm, 21 (2), pp. 35–38. https://doi.org/10.1145/3131214.3131225

[8] Ziuziun, V., Kulkovets, V., & Parasiuk, L. (2024). Development of a Decision Support Information System for Managing Large Agile Teams in IT Projects. Collection of Scientific Papers of Admiral Makarov National University of Shipbuilding, 4(497), pp. 166-172, https://doi.org/10.15589/znp2024.4(497).23

[9] Ziuziun, V., & Petrenko, N. (2025). AI Solutions for Optimizing SCRUM: Predicting Team Performance. Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies, (2) 14, pp. 85-89. https://doi.org/10.20998/2079-0023.2025.02.11

[10] Ziuziun, V. & Petrenko, N. (2025). AI-Enhanced System Design for Agile Sprint Management and Velocity Prediction. 2025 IEEE 5th International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, pp. 1-6, https://doi.org/10.1109/SIST61657.2025.11139278

[11] Ziuziun, V., & Avramets, A. (2025). Model' ta kryterij ocinky efektyvnosti intelektual'noi' fil'tracii' spovishhen' dlja IT-fahivciv. [Model and evaluation criterion for the effectiveness of intelligent notification filtering for IT specialists]. Information Technology and Society, 4 (19).

[12] Carzaniga, A. (2025). Elements of Queuing Theory. URL: https://www.inf.usi.ch/carzaniga/edu/adv-ntw/queuing-theory-notes.pdf

[13] Needham M., & Hodler, A. E. (2019). Graph Algorithms. Practical Examples in Apache Spark and Neo4j. Retrieved from: https://web4.ensiie.fr/~stefania.dumbrava/GraphAlgorithms.pdf

[14] Wilson, R. J. (1996). Introduction to Graph Theory. Retrieved from: https://webhomes.maths.ed.ac.uk/~v1ranick/papers/wilsongraph.pdf

[15] (2024). An Introduction to Graph Theory. Retrieved from: https://www.datacamp.com/tutorial/introduction-to-graph-theory

[16] Sharma, S., & Kumar, V. A. (2022). Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future. Arch Computat Methods Eng 29, 5605–5633. https://doi.org/10.1007/s11831-022-09778-9

[17] Sharifi, M. R., Akbarifard, S., & Qaderi, K. *et al.* (2021). A new optimization algorithm to solve multi-objective problems. *Sci Rep* 11, 20326. https://doi.org/10.1038/s41598-021-99617-x

[18] Lapeyre, B. (2021). Monte Carlo Methods and Stochastic Algorithms. Retrieved from: https://cermics.enpc.fr/~bl/monte-carlo/poly.pdf

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Parnin, C., Gorg, C., & Rugaber, S. (2009). TaskBoard: Tracking pertinent task artifacts and plans. 2009 IEEE 17th International Conference on Program Comprehension, Vancouver, BC, Canada, pp. 317-318, https://doi.org/10.1109/ICPC.2009.5090076

[2] Parnin, C., Gorg, C., & Rugaber, S. (2010). CodePad: interactive spaces for maintaining concentration in programming environments. In Proceedings of the 5th international symposium on Software visualization (SOFTVIS '10). Association for Computing Machinery, New York, NY, USA, 15–24. https://doi.org/10.1145/1879211.1879217

[3] Parnin, C., & Rugaber, S. (2012). Programmer information needs after memory failure. 2012 20th IEEE International Conference on Program Comprehension (ICPC), Passau, Germany, 123-132, https://doi.org/10.1109/ICPC.2012.6240479.

[4] Mehrotra, A., & Musolesi, M. (2018). Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges. Computer Science. Human-Computer Interaction. https://doi.org/10.48550/arXiv.1711.10171

[5] Bazinette, V., Cohen, N. H., & Ebling M. R., et al. (2001). An Intelligent Notification System. IBM Research Report. URL: https://www.researchgate.net/publication/228970929_An_intelligent_notification_system

[6] Mehrotra, A., Hendley, R., & Musolesi, M. (2016). PrefMiner: mining user's preferences for intelligent mobile notification management. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16). Association for Computing Machinery, New York, NY, USA, 1223–1234. https://doi.org/10.1145/2971648.2971747

[7] Mehrotra, A., Hendley, R., & Musolesi, M. (2017). Interpretable Machine Learning for Mobile Notification Management: An Overview of PrefMiner. GetMobile: Mobile Comp. and Comm, 21 (2), 35–38. https://doi.org/10.1145/3131214.3131225

[8] Ziuziun, V., Kulkovets, V., & Parasiuk, L. (2024). Development of a Decision Support Information System for Managing Large Agile Teams in IT Projects. Collection of Scientific Papers of Admiral Makarov National University of Shipbuilding, 4(497), 166-172, https://doi.org/10.15589/znp2024.4(497).23

[9] Ziuziun, V., & Petrenko, N. (2025). AI Solutions for Optimizing SCRUM: Predicting Team Performance. Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies, (2) 14, 85-89. https://doi.org/10.20998/2079-0023.2025.02.11

[10] Ziuziun, V. & Petrenko, N. (2025). AI-Enhanced System Design for Agile Sprint Management and Velocity Prediction. 2025 IEEE 5th International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, pp. 1-6, https://doi.org/10.1109/SIST61657.2025.11139278

[11] Зюзюн, В., & Аврамець, А. (2025). Модель та критерій оцінки ефективності інтелектуальної фільтрації сповіщень для IT-фахівців. Інформаційні технології та суспільство, 4 (19).

[12] Carzaniga, A. (2025). Elements of Queuing Theory. URL: https://www.inf.usi.ch/carzaniga/edu/adv-ntw/queuing-theory-notes.pdf

[13] Needham M., & Hodler, A. E. (2019). Graph Algorithms. Practical Examples in Apache Spark and Neo4j. URL: https://web4.ensiie.fr/~stefania.dumbrava/GraphAlgorithms.pdf

[14] Wilson, R. J. (1996). Introduction to Graph Theory. URL: https://webhomes.maths.ed.ac.uk/~v1ranick/papers/wilsongraph.pdf

[15] (2024). An Introduction to Graph Theory. URL: https://www.datacamp.com/tutorial/introduction-to-graph-theory

[16] Sharma, S., & Kumar, V. A. (2022).Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future. Arch Computat Methods Eng 29, pp. 5605–5633. https://doi.org/10.1007/s11831-022-09778-9

[17] Sharifi, M. R., Akbarifard, S., & Qaderi, K. *et al.* (2021). A new optimization algorithm to solve multi-objective problems. *Sci Rep* 11, 20326. https://doi.org/10.1038/s41598-021-99617-x

[18] Lapeyre, B. (2021). Monte Carlo Methods and Stochastic Algorithms. URL: https://cermics.enpc.fr/~bl/monte-carlo/poly.pdf

Дата першого надходження статті до видання: 22.01.2026
Дата прийняття статті до друку після рецензування: 17.02.2026
Дата публікації (оприлюднення) статті: 22.05.2026