УДК 519.2
DOI https://doi.org/10.24144/2616-7700.2026.49(2).161-164

О. С. Веретьонкін

Київський національний університет ім. Т. Шевченка,
аспірант кафедри прикладної статистики
alexveretenkin@gmail.com
ORCID: https://orcid.org/0009-0007-4776-5333

МОДЕЛЬ ОПТИМАЛЬНОГО КЕРУВАННЯ ТЕХНІЧНИМ БОРГОМ У ПРОЦЕСАХ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Технічний борг є одним із ключових чинників, що визначають довгострокову підтримуваність, швидкість еволюції та вартість супроводу програмних систем. Попри значну кількість досліджень, присвячених класифікації, вимірюванню та практикам управління технічним боргом, формалізовані моделі його динаміки залишаються недостатньо розробленими. У статті запропоновано компактну математичну модель керування технічним боргом у процесі розробки програмного забезпечення. Процес розробки інтерпретується як динамічна система з двома взаємопов'язаними потоками: реалізацією нової функціональності та усуненням технічного боргу. Враховано вплив накопиченого боргу на ефективну продуктивність команди, сформульовано умови стійкості системи та отримано оцінку частки ресурсів, яку доцільно спрямовувати на його погашення. Для практичної ілюстрації використано відкриті дані GitHub Issues для репозиторію Angular. Показано, що навіть спрощена інтерпретація моделі дає змогу виявляти тенденції до накопичення backlog і формувати кількісно обґрунтовані орієнтири для розподілу ресурсів у процесі розробки.

Ключові слова: технічний борг, стохастичне моделювання, оптимальне керування, GitHub Issues, backlog, рефакторинг.

1. Вступ. Технічний борг є важливим чинником, що визначає довгострокову підтримуваність і продуктивність програмних систем. Він виникає як наслідок компромісів між швидкістю розробки та якістю технічних рішень і з часом призводить до зростання вартості супроводу, ускладнення архітектури та сповільнення розвитку системи [1–3]. За оцінками McKinsey, суттєва частка ресурсів, призначених для розвитку цифрових продуктів, фактично витрачається на подолання наслідків накопиченого технічного боргу [8].

У сучасних дослідженнях технічний борг аналізується з позицій класифікації, практик управління, архітектурних рішень та емпіричного виявлення [4–7]. Однак формальні моделі, які описують його динаміку та дозволяють обґрунтовано визначати розподіл ресурсів між розробкою нової функціональності й рефакторингом, представлені обмежено.

Метою статті є побудова компактної моделі керування технічним боргом у процесі розробки програмного забезпечення, визначення умов стійкості такої системи та демонстрація можливості практичної інтерпретації моделі на основі відкритих даних GitHub.

2. Основний результат. У роботах Brown et al. [2] та Kruchten, Nord, Ozkaya [3] технічний борг розглядається як об'єкт системного керування, а не лише як метафора. Систематичний огляд Li, Avgeriou та Liang [4] показує, що більшість досліджень зосереджені на типології боргу, практиках його виявлення

та оцінювання. Емпіричні праці Ernst et al. [5] і Yli-Huumo et al. [6] засвідчують, що управління технічним боргом у командах є важливою, але недостатньо формалізованою практикою.

Окрему увагу дослідники приділяють архітектурним рішенням як джерелу довгострокового технічного боргу [7]. Це підтверджує доцільність побудови простих моделей, які дозволяють пов'язати накопичення боргу, продуктивність команди та політику розподілу ресурсів у межах єдиного формального опису.

2.1. Математична модель. Нехай $F(t)$ — кількість невиконаних feature-заявок у момент часу t, $D\,(t) \geq 0$ — обсяг накопиченого технічного боргу, $\lambda > 0$ — інтенсивність надходження нових задач, $\mu > 0$ — базова продуктивність команди. Позначимо через $u(t)$, $0 \leq u\,(t) \leq 1$, частку ресурсів, що спрямовується на усунення технічного боргу. Тоді частка $1 - u(t)$ використовується для реалізації нової функціональності.

Припустимо, що виконання feature-задач породжує технічний борг із середнім коефіцієнтом $\alpha > 0$, а накопичений борг знижує ефективну продуктивність команди відповідно до функції $g\,(D) = \exp\,(-kD)$, $k > 0$. За цих припущень динаміка системи описується рівняннями

$$\begin{cases} \frac{dF}{dt} = \lambda - (1-u)\mu\,\exp(-kD); \\ \frac{dD}{dt} = \alpha(1-u)\mu\,\exp(-kD) - u\mu. \end{cases}$$

Для формалізації задачі керування введемо функціонал витрат

$$J\,(u) = \int\limits_0^T \left(c_1 F\,(T) + c_2 D\,(T)\right) dt,$$

де $c_1 > 0$ та $c_2 > 0$ задають вагу затримки реалізації функціональності та накопичення технічного боргу відповідно.

2.2. Умови стійкості та розподіл ресурсів. Із першого рівняння випливає умова стійкості системи:

$$\lambda < (1-u)\,.$$

Отже, середня швидкість надходження задач не повинна перевищувати ефективну швидкість їх виконання. Інакше backlog зростатиме необмежено, а система працюватиме в режимі перевантаження. У стаціонарному режимі з балансу генерації та погашення технічного боргу можна отримати оцінку частки ресурсів, яку доцільно спрямовувати на рефакторинг:

$$u^* \approx \frac{\alpha\lambda}{\mu + \alpha\lambda}.$$

Це співвідношення показує, що зі зростанням інтенсивності надходження нових задач або швидкості генерації боргу частка ресурсів на його усунення має зростати. Натомість підвищення продуктивності команди дає змогу зменшити u^* без втрати стійкості системи.

3. Емпірична ілюстрація. Для практичної апробації моделі використано відкриті дані GitHub Issues. Нехай $A(t)$ — кількість створених задач за період t,

$C(t)$ — кількість закритих задач, $B(t)$ — накопичений backlog. Тоді емпірична динаміка описується співвідношенням:

$$B\,(t+1) = B\,(T) + A\,(T) - c\,(T)\,.$$

На основі часових рядів для репозиторію Angular отримано наближені оцінки $\lambda \approx 134.69$ задач/місяць та $\mu \approx 113.38$ задач/місяць, що вказує на перевантажений режим функціонування. Для підмножини задач, інтерпретованих як технічний борг, отримано оцінку $\lambda_{td} \approx 4.94$ задач/місяць. Якщо в межах спрощеної інтерпретації покласти $\alpha = 1$, тоді

$$u^* \approx \frac{4.94}{113.38 + 4.94} \approx 0.042.$$

Отже, модельна оцінка вказує на доцільність спрямування близько 4–5% ресурсів команди на регулярне усунення технічного боргу. Цю величину слід розглядати як орієнтир першого наближення, а не як жорстку нормативну вимогу.

4. Висновки. Запропоновано компактну модель керування технічним боргом, яка дозволяє формалізувати взаємозв'язок між потоком нових задач, накопиченням технічного боргу та політикою розподілу ресурсів команди. У моделі сформульовано умову стійкості системи та отримано оцінку частки ресурсів, яку доцільно спрямовувати на рефакторинг.

Практична ілюстрація на основі GitHub Issues показала, що навіть спрощена емпірична інтерпретація моделі може бути корисною для виявлення тенденцій до накопичення backlog і підтримки рішень щодо керування ресурсами в процесі розробки програмного забезпечення.

Конфлікт інтересів

Автор заявляє, що немає конфлікту інтересів щодо даного дослідження, включаючи фінансовий, особистий, авторський або будь-який інший, який міг би вплинути на дослідження, а також на результати, представлені в даній статті.

Фінансування

Дослідження було проведено без фінансової підтримки.

Доступність даних

Усі дані доступні в цифровій або графічній формі в основному тексті рукопису.

Використання штучного інтелекту

Під час підготовки статті інструменти штучного інтелекту використовувалися як допоміжний засіб, тоді як постаноска та формулювання задачі, основні етапи розв'язання, аналіз даних, інтерпретація результатів та остаточна перевірка та валідація здійснювалися автором.

Список використаної літератури

1. Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., & Zazworka, N. (2010). Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research.* (pp. 47–52). https://doi.org/10.1145/1882362.1882373
2. Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software,* 29(6), 18–21. https://doi.org/10.1109/MS.2012.167
3. Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software,* 101, 193–220. https://doi.org/10.1016/j.jss.2014.12.027
4. Dalal, V., Krishnakanthan, K., Munstermann, B., & Patenge, R. (2020). Tech debt: Reclaiming tech equity. McKinsey Digital. Retrieved from https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Tech%20debt%20Reclaiming%20tech%20equity/Tech-debt-Reclaiming-tech-equity.pdf
5. Soliman, M., Avgeriou, P., & Li, W. (2021). Architectural design decisions that incur technical debt: An industrial case study. *Information and Software Technology,* 137, Article 106669. https://doi.org/10.1016/j.infsof.2021.106669
6. Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016). How do software development teams manage technical debt? An empirical study. *Journal of Systems and Software,* 120, 195–218. https://doi.org/10.1016/j.jss.2016.05.018

Veretonkin O. S. Optimal Control Model of Technical Debt in Software Development Processes.

Technical debt is one of the key factors that determine the long-term maintainability, evolution speed, and support cost of software systems. Despite a considerable number of studies devoted to the classification, measurement, and management of technical debt, formalized models of its dynamics remain insufficiently developed. This paper proposes a compact mathematical model for technical debt management in software development processes. The development process is interpreted as a dynamic system with two interrelated flows: implementation of new functionality and technical debt reduction. The model takes into account the impact of accumulated debt on the effective productivity of the development team, formulates system stability conditions, and provides an estimate of the share of resources that should be allocated to technical debt reduction. For practical illustration, open GitHub Issues data from the Angular repository are used. The results show that even a simplified interpretation of the model makes it possible to identify backlog accumulation trends and derive quantitatively grounded guidelines for resource allocation in the development process.

Keywords: technical debt, stochastic modeling, optimal control, GitHub Issues, backlog, refactoring.

Отримано: 25.03.2026 Прийнято: 10.04.2026 Опубліковано: 30.04.2026