

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
АКАДЕМИЯ ПРОБЛЕМ КАЧЕСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
РОССИЙСКАЯ АКАДЕМИЯ КОСМОНАВТИКИ им. К. Э. ЦИОЛКОВСКОГО
ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР «ИНФОРМАТИКА И УПРАВЛЕНИЕ» РАН
(ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР РАН имени А. А. ДОРОДНИЦЫНА)
АО «УПКБ "ДЕТАЛЬ"», АО «РАДИОЗАВОД»
АО «ФНПЦ «ПО "СТАРТ" имени М. В. ПРОЦЕНКО», АО «НИИЭМП»
АО «ПНИЭИ», АО «НИИФИ», АО «НПП "РУБИН"»
ЕВРАЗИЙСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ имени Л. Н. ГУМИЛЕВА
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ



ТРУДЫ
МЕЖДУНАРОДНОГО СИМПОЗИУМА
НАДЕЖНОСТЬ И КАЧЕСТВО

Том 1

*Посвящается году единства народов России и 65-летию
со дня первого полета Юрия Гагарина в космос*

ПЕНЗА 2026

Труды Международного симпозиума НАДЕЖНОСТЬ И КАЧЕСТВО. – 2026. – Т. 1. – 467 с.

Представлены доклады XXXI Международного симпозиума «Надежность и качество», посвященного году единства народов России и 65-летию со дня первого полета Юрия Гагарина в космос, проходившего с 25 по 31 мая 2026 г. в г. Пензе. Рассмотрены актуальные проблемы теории и практики повышения надежности и качества; эффективности внедрения инновационных и информационных технологий в фундаментальные научные и прикладные исследования, образовательные и коммуникативные системы и среды, экономику и юриспруденцию; методов и средств анализа и прогнозирования показателей надежности и качества приборов, устройств и систем, а также анализа непараметрических моделей и оценки остаточного ресурса изделий двойного назначения; ресурсосбережения; проектирования интеллектуальных экспертных и диагностических систем, систем управления и связи, интерактивных, телекоммуникационных сетей и сервисных систем; экологического мониторинга и контроля состояния окружающей среды и биологических объектов; исследования физико-технологических процессов в науке, технике и технологиях для повышения качества выпускаемых изделий радиопромышленности, приборостроения, аэрокосмического и топливно-энергетического комплексов, электроники и вычислительной техники и др.

Издание предназначено для студентов, аспирантов, ученых и научных сотрудников, заинтересованных в исследовании надежности и качества эффективных комплексных систем.

УДК 621.396.6:621.315.616.97:658:562

Оргкомитет благодарит за поддержку в организации и проведении Международного симпозиума и издания настоящих трудов Министерство науки и высшего образования РФ, Правительство Пензенской области, Академию проблем качества РФ, Российскую академию космонавтики имени К. Э. Циолковского, Российскую инженерную академию, ФИЦ ИУ РАН (Вычислительный центр РАН имени А. А. Дородницына), АО «УПКБ “ДЕТАЛЬ”», АО «НИИФИ», АО «ПНИЭИ», АО «НПП “РУБИН”», АО «РАДИОЗАВОД», АО «ППО “ЭЛЕКТРОПРИБОР”», АО «НИИЭМП», АО «ФНПЦ «ПО “СТАРТ” имени М. В. Проценко», АО «НИКИРЭТ», ООО «Экспериментальная мастерская “Наука-Софт”», Евразийский национальный университет имени Л. Н. Гумилева, Пензенский государственный университет.

Издание зарегистрировано в Российском индексе научного цитирования (РИНЦ) с 2005 г.

Редакционная коллегия:

Юрков Н. К. – главный редактор;
Трусов В. А. – ответственный секретарь;
Гуляков А. Д., Васин С. М., Сыдыков Е. Б., Абрамов О. В.,
Артамонов Д. В., Безродный Б. Ф., Бецков А. В., Дивеев А. И., Иофин А. А.,
Кашищанов В. А., Кемалов Б. К., Острейковский В. А., Пиганов М. Н.,
Роберт И. В., Романенко Ю. А., Савченко В. П., Садыхов Г. С.,
Халютин С. П., Шахнов В. А., Якимов А. Н.

УДК 004.4'24

Timonin¹ A.Yu., Elamrani² B.

^{1,2} Penza State University, Penza, Russia

¹ timonin@pnzgu.ru

² badrelamrani33@gmail.com

VIBE CODING APPLICABILITY TO REFACTOR EXISTING SOFTWARE CODE

Abstract. The potential of using modern neural methods and models based on large language models is explored to automate the writing of updated software code when preparing its new versions. It also highlights the advantages and disadvantages of neural code generation methods compared to traditional coding tools, where the software engineer inspired by his past experiences or utilizes different ready-made templates and algorithms.

Keywords: artificial intelligence, code refactoring, large language models, programming automation, vibe coding

Тимонин¹ А.Ю., Эль Амрани² Б.

^{1,2} Пензенский государственный университет, Пенза, Россия

¹ timonin@pnzgu.ru

² badrelamrani33@gmail.com

ПРИМЕНИМОСТЬ ВАЙБ-КОДИНГА ДЛЯ ПЕРЕПИСЫВАНИЯ УЖЕ СУЩЕСТВУЮЩЕГО ПРОГРАММНОГО КОДА

Аннотация. Исследуется потенциал применимости современных нейросетевых методов и моделей на основе больших языковых моделей для автоматизации написания обновленного кода программного обеспечения при подготовке его новых версий. Также подчеркиваются преимущества и недостатки методов нейронной генерации кода по сравнению с традиционными инструментами программирования, где разработчик программного обеспечения вдохновляется своим прошлым опытом или использует различные готовые шаблоны и алгоритмы.

Ключевые слова: искусственный интеллект, рефакторинг кода, большие языковые модели, автоматизация программирования, вайб-кодинг

The development of generative neural networks has become a trend of the current decade. Existing artificial intelligence (AI) projects already enable the generation of logically structured texts, audiovisual images, and their combinations on given topics [1]. Among the promising areas of AI application is the automation of software development, including refactoring (updating, rewriting, and optimizing code) of existing software products [2]. The implementation of such development methods is being carried out both professionally at large companies (MTS, Sber, Yandex, etc.), and privately by various developers, including university students, during practical, laboratory, coursework, and theses.

The term "vibe coding" refers to a programming method that uses large language models and artificial intelligence to generate code based on a natural language description of a task. It was first officially coined in February 2025 by machine learning scientist Andrej Karpathy [3]. The word "vibe" in English refers to a specific atmosphere, a mood, emanating from something or someone. The method itself began to be used when GPT-like neural networks began to be trained on a variety of open-source computer program source codes.

At its core, this is the automatic generation of program code, based not on the developer's experience or the composition/decomposition of existing algorithms for standard tasks, but on the neural network's attempt to match a verbal description of a given task with existing examples of similar solutions. Vibe coding methods are an evolutionary development of programmers' approaches to automating software development and simplifying human-machine interaction.

The first milestone in the automation of information tasks can be considered the advent of programmable computers and the first algorithmic languages. Subsequently, programming languages evolved, and a division into low- and high-level languages occurred:

- more understandable for the computer and having more possibilities for code optimization and efficient use of computer hardware;
- more understandable for the developer, which allowed for faster program development, opportunities for porting software to various hardware platforms, etc.

Then, coding tools began to evolve. Specialized IDEs began supporting cross-compilation, combining code from different languages, and the ability to automatically check for errors and optimize code during compilation/translation. The use of extensible libraries and programming language modules expanded. Python, with its extensive library of extensions, became particularly prominent, making it popular in various fields, from programming education to cutting-edge developments in data science and machine learning.

The next important step in the automation of software development was the development of visual programming tools, which were initially created to teach the basics of programming (the LOGO language as an example), but later became part of development environments for creating graphical interfaces, UML diagrams, and data schemas.

Beyond the technical prerequisites for the emergence of vibe coding, there are also social and communicative factors. Initially, developer communities were largely closed, limited to the organizations in which developers worked, and to the academic environment. With the development of the internet, opportunities for exchanging ideas about algorithms and their implementations in various programming languages became greater. Communities of developers engaged in the collective development of open-source software began to develop. In addition to information and reference resources dedicated to programming, various programming community websites (such as cyberforum.ru, github.com, stackoverflow.com) emerged. Initially, their audience consisted of professionals, but with the increasing accessibility of the internet, the main audience of such sites became students of technical universities and novice developers. Since the mid-2010s, due to the popularity of such sites and the richness of their code bases, code copying has become widespread. The development of the concept of free and open-source software, where source code is published in the public domain and its development is carried out collectively by a community of developers, also contributed.

During the training of the latest generative networks based on the above-mentioned resources, all the existing experience of developers was preserved within the artificial intelligence models, allowing any user of these models to access it without any initial technical training, which has led to the popularity of vibe coding.

The increasing need to rewrite (refactor) existing software code to keep it up-to-date is driven by a number of factors. Various software libraries, modules, and dependencies are constantly being updated, vulnerabilities are being identified, and new functionality needs to be implemented in existing software products. New programming languages, such as Rust, Go, and Mojo, are gaining popularity. They offer improved handling of dynamically allocated memory, parallelism, garbage collection, testability, and support for third-party user libraries. This requires developers to invest additional resources in reading existing code and identifying problematic areas, especially if the code was written long ago, in a different language or an older version of the same language, or if the original author no longer maintains it. There are also situations where source code is corrupted or lost.

Rewriting costs must be reduced, ideally by automating them. A large project example is the Windows operating system, which contains a huge amount of legacy code from previous versions dating back to the 1990s. Keeping code up-to-date is a major challenge even for a corporation as large as Microsoft, as updating system components takes a significant amount of time and often leads to difficult-to-trace unexpected errors. Another example is the Linux kernel, whose codebase is growing exponentially and has already exceeded the ability of a single person to study it. Despite strict guidelines for accepting changes, the process of rewriting individual modules from C to Rust is slow and controversial within the community due to the disproportionate labor costs and the theoretical benefit of leveraging the advantages of the new language. The development of cross-platform software and support for multiple devices on the ARM architecture [4] also highlights the need for automated software development, which can be facilitated by vibe coding.

The advantages of vibe coding include reducing the labor costs of routine, complex, and repetitive work, allowing developers to focus on the task itself rather than the technical details of its implementation. It also allows developers to quickly prepare several possible solutions, select the best one, and then work exclusively with it. Another advantage of vibe coding is the ability for professional programmers to quickly get started with an unfamiliar programming language.

However, not only experienced developers but also beginners use vibe coding. This poses the following risks:

- due to the lack of practical experience in developing complex software, they are unable to understand and manually refine the generated code;
- they do not develop skills in algorithmization, independent coding, and teamwork.

Because of this, significant errors may be missed, and the code may be suboptimal and contain hidden vulnerabilities. Neural networks often act erratically, generating seemingly plausible but meaningless or incorrect code that can be difficult to recognize even for an experienced programmer. This problem can be solved by introducing methods for extracting semantics [5] from the generated code and comparing it with the semantics of the original code. Also, neural network error detection methods, similar to speech error recognition [6], can be used as feedback mechanisms within a neural network to self-check its outputs. It's worth noting that some language models are trained not only on open-source projects, which are generally prone to vulnerabilities and backdoors, but also on proprietary projects protected by trade secrets. This creates legal threats for developers. A particular example is ReactOS, whose developers are required to remain unaware of the Windows OS source code. Furthermore, the laws of some countries prohibit disassembling third-party software code, including for informational purposes.

Vibe coding can be considered a new stage in software development automation. In the future, programmers' work will change somewhat: instead of developing detailed algorithms from scratch and writing code tailored to the specifics of a programming language, they will automatically assemble code components from generated templates into a coherent system, selecting the most optimal ones for their tasks from a variety of proposals. To achieve this, developers will need to be able to work with other people's code, resolve dependency conflicts in variable naming, data type selection, and the number of arguments passed, and verify source code for adequacy. Vibe coding methods can reduce the cost of "grunt" work typically assigned to interns and junior developers; the creation of minimum acceptable code, test projects, and simple software modules will be entirely transferred to artificial intelligence. This will raise the minimum requirements for developers' professional competencies and skills, meaning that the work of middle and senior developers will essentially remain entirely relevant.

REFERENCES

1. Key capabilities of generative systems for creating educational materials // Sber University. URL: <https://courses.sberuniversity.ru/ai-education/5/1> (In Russ.)
2. Kulikov A.A., Shuralev S.I. Design of a neural network for code refactoring // BBK 1 H 34. - P. 5735. (In Russ.)
3. Edwards B. Will the future of software development run on vibes? // Ars Technica URL: <https://arstechnica.com/ai/2025/03/is-vibe-coding-with-ai-gnarly-or-reckless-maybe-some-of-both/>
4. Timonin A.Yu., Makeev D.I., Filippov O.R. Difficulties in developing free system software for devices on the ARM architecture // Novye informacionnye tekhnologii i sistemy (NITiS-2023) : Sbornik nauchnyh statej po materialam XX Mezhdunarodnoj nauchno-tekhnicheskoj konferencii (New information technologies and systems (NITiS-2023): Collection of scientific articles based on the materials of the XX International scientific and technical conference), Penza, November 16-17, 2023. - Penza: Penza State University, 2023. - P. 191-195. - EDN BXSVMC. (In Russ.)
5. Gordeeva O.A., Kudrina M.A. Automated system for semantic analysis of small text // Trudy mezhdunarodnogo simpoziuma "Nadezhnost' i kachestvo" (Proceedings of the international symposium "Reliability and quality"). - 2025. - V. 1. - P. 415-422. - EDN OWIDOV. (In Russ.)
6. Alimuradov A.K., Berezina A.O., Medvedev A.V. Development of a neural network method for detecting speech errors // Trudy mezhdunarodnogo simpoziuma "Nadezhnost' i kachestvo" (Proceedings of the international symposium "Reliability and quality"). - 2025. - V. 1. - P. 320-323. - EDN DYPMZM. (In Russ.)