Milan Wowra
Wydział Fizyki UW

vibe coding – puszka Pandory, czy Święty Graal programistów?

Obecnie już trudno spotkać człowieka, który nie miał styczności lub nie słyszał o „sztucznej inteligencji", czyli dokładniej z dużymi modelami językowymi, często potocznie nazywanymi LLM-ami od ich angielskiego skrótowca. Chyba każdy z nas wpisał chociaż raz jakieś pytanie do ChataGPT i z radością patrzył jak na ekran wylewają się kolejne linijki mniej lub bardziej sensownej odpowiedzi na nasz problem. Jednym z częściej spotykanych zastosowań jest używanie takich narzędzi w trakcie pisania programów komputerowych. Modele językowe uczone na setkach milionów linii kodu źródłowego rozmaitych programów, radzą sobie całkiem dobrze w takich zadaniach (a przynajmniej sprawiają takie wrażenie). Nic dziwnego, że to programiści jako jedni z pierwszych zaczęli na dużą skalę stosować AI w swojej pracy, częściowo do pomocy w pisaniu programów, szukania błędów lub tłumaczenia działania napotkanego kodu. W miarę z zwiększaniem się złożoności modeli możliwe stało się tworzenie praktycznie całych, kompletnych programów komputerowych lub stron internetowych za pomocą kilku (lub kilkunastu) tzw. „promptów" czyli zapytań do AI. Umożliwiło to „pisanie" programów komputerowych posługując się językiem naturalnym, a nie będąc skrępowanym sztywnymi wymogami składni języka programowania. Zjawisko to znalazło swoją nazwę – „vibe coding". Nazwy tej prawdopodobnie jako pierwszy użył jeden z byłych pracowników OpenAI - Andrej Karpathy w lutym 2025 roku w swoim tweecie na platformie X (dawniej Twitter) [1].

Można się zastanawiać co w tym takiego ważnego, ale warto zwrócić uwagę, że często główną trudnością tworzenia programu było wymyślenie „jak to ma działać", a potem ubranie tego pomysłu w odpowiednie struktury i składnię języka, w którym tworzyło się program. Przy użyciu modeli językowych wystarczy opisać co się chce otrzymać, a na ekranie pojawia się rozwiązanie. Nie zawsze idealne i nie od razu dokładnie takie jakbyśmy chcieli, ale po kilku próbach, doprecyzowaniu swojego żądania i wklejeniu kilku komunikatów o błędach do konwersacji z modelem, istnieje duża szansa na to, że otrzyma się coś, co odpowiada pierwotnym wymaganiom. W swoim tweecie Karpathy pisze, że można „zapomnieć, że kod istnieje", a komunikaty błędów „wkleić bez komentarza" i skopiować rozwiązanie – czyli, czy to święty Graal programistów, który rozwiązuje wszystkie problemy?

Trzeba zwrócić uwagę na kilka aspektów obecnie tworzonych programów i systemów komputerowych, przede wszystkim na kwestie takie jak ich poprawność działania, wydajność, niezawodność i bezpieczeństwo. Nie są to kwestie szczególnie istotne dla programów tworzonych hobbystycznie, na swoje

własne potrzeby lub dla zaspokojenia ciekawości – w końcu świat się nie zawali jeśli nasz program naśladujący kalkulator jako wynik działania „2 + 3" poda „23", jednak w większości profesjonalnych zastosowań jest to wynik nie do zaakceptowania. W sytuacji, gdy nie zwracamy uwagi na kod programu to pozbawiamy się możliwości weryfikacji tego, czy program rzeczywiście będzie działał tak jak chcemy, oraz w sposób który rozumiemy, w każdym z przypadków. Otwiera to także drogę do powstawania drobnych, z pozoru nieistotnych, „uchybień" w kodzie, których świadomy programista by nie popełnił, a które mogą stać się drogą do incydentów bezpieczeństwa. Raport Wiz inc. wskazuje, że w około 1 na 5 aplikacjach tworzonych za pomocą „vibe codingu" znajdują się krytyczne błędy bezpieczeństwa lub błędy konfiguracji [2, 3]. Przykładami takich „uchybień" przy użyciu modeli językowych do programowania jest: przechowywanie kluczy API po stronie klienta, brak walidacji wejścia, logika autoryzacji po stronie klienta oraz używanie przestarzałych, zewnętrznych bibliotek [3]. Czyli „vibe coding" to jednak puszka Pandory?

Warto w tym miejscu przytoczyć wypowiedź Linusa Torvaldsa, twórcy Git-a i jądra Linuxa. Stwierdza on, że jest „stosunkowo pozytywnie" nastawiony do używania „vibe codingu", ale nie w stosunku do jądra Linuxa, czyli kodu dla którego poprawność, wydajność i bezpieczeństwo działania jest krytyczne. Jednocześnie może to być dla kogoś sposób do „skłonienia komputera do zrobienia czegoś, czego inaczej ktoś może nie potrafiłby zrobić". Zauważa też, że „może to być bardzo, bardzo zły pomysł z punktu widzenia utrzymania oprogramowania" [4].

Historia, jak i poprawna frazeologicznie fortuna, kołem się toczy. Można odnieść wrażenie, że „vibe coding" spełnia funkcję podobną jak język programowania BASIC we wczesnych latach informatyki. Miał on umożliwić łatwe pisanie programów osobom, które nie znały się tak bardzo na programowaniu, aby móc pisać je bezpośrednio w assemblerze, a programistom dostarczać wysokopoziomowy, łatwy do zrozumienia i wydajny język. Można było znaleźć gorących orędowników BASIC-a, jak i jego przeciwników wskazujących złe praktyki jakich uczył BASIC, takie jak na przykład wszechobecne używanie instrukcji „GOTO". Podobnie obecny „vibe coding" programistom pozwala szybciej i wygodniej „pisać" kod, otwiera drzwi informatyki dla osób, które nie miały wcześniej z nią zbyt dużej styczności, ale też niestety wiąże się to, często z nieświadomym umieszczaniem potencjalnych podatności bezpieczeństwa w kodzie.

Czy zatem „vibe coding" to puszka Pandory, czy Święty Graal? Jednoznacznej odpowiedzi nie da się udzielić, tak samo jak na pytanie „czy foton jest cząstką, czy falą?" – jednym i drugim, zależy to od doświadczenia. Na pewno jest to zjawisko, na które jest miejsce w ciągle prężnie rozwijającym się świecie informatyki. Znajduje to potwierdzenie w statystykach udostępnionych przez Y Combinator – jednego z akceleratorów z doliny krzemowej, który ogłosił, że w

około 25% startupów w ich portfolio W25, ponad 95% właściwego kodu produktów powstało przez „vibe coding” [5].

Na końcu warto zauważyć, że niestety jak i „42” nie jest spodziewaną ostateczną odpowiedzią na „wielkie pytanie o życie, wszechświat i całą resztę” tak samo „vibe coding” nie jest ostatecznym rozwiązaniem problemu tworzenia oprogramowania i udziału ludzi w tym procesie. Trzeba zauważyć, że technologia AI na obecnym poziomie rozwoju jest raczej narzędziem człowieka w tworzeniu rzeczywistości, a nie autonomicznym tworem zdolnym do samodzielnego działania.

Źródła:

[1] https://web.archive.org/web/20250306124303/https://arstechnica.com/ai/2025/03/is-vibe-coding-with-ai-gnarly-or-reckless-maybe-some-of-both/ (dostęp 03.01.2026, kopia z 06.03.2025)
[2] https://www.wiz.io/blog/common-security-risks-in-vibe-coded-apps (dostęp 03.01.2026)
[3] https://www.kaspersky.com/blog/vibe-coding-2025-risks/54584/ (dostęp 03.01.2026)
[4] https://www.theregister.com/2025/11/18/linus_torvalds_vibe_coding/ (dostęp 03.01.2026)
[5] https://techcrunch.com/2025/03/06/a-quarter-of-startups-in-ycs-current-cohort-have-codebases-that-are-almost-entirely-ai-generated/ (dostęp 03.01.2026)