



**RAQAMLI TA’LIM MUHITIDA BO’LAJAK DASTURCHI
MUHANDISLARNI TAYYORLASHDA KOD SIFATINI
BAHOLASHNING INNOVATSION USULLARI:
MUTATSION TESTLASH VA SUN’IY INTELLEKT
MODELLARI INTEGRATSIYASI**

Xamrayev Xurshidbek Komiljon o’g’li
UBS “Innovatsion texnologiyalar”
kafedrası stajor-o’qituvchisi

Annotatsiya: Raqamli ta’lim muhitida dasturlash bo’yicha bilim va ko’nikmalarni shakllantirishda kod sifatini samarali baholash muhim o’rin tutadi. An’anaviy qo’lda tekshirish va avtomatlashtirilgan testlar talabalarning individual xatolarini aniqlash va ta’limiy qayta aloqa (feedback) berishda yetarlicha mufassal bo’lmaydi. Ushbu maqola ikki ilg’or yondashuv – mutatsion testlash va sun’iy intellekt modellarini integratsiyalash asosida kod sifatini baholashning innovatsion usullarini tahlil qiladi. Mutatsion testlash sinov to’plamlarining samaradorligini sintetik nuqsonlar (mutatsiyalar) orqali o’lchasa, katta til modellari (SI) esa manba kodini tushunib, shaxsiylashtirilgan tavsiyalar bera oladi. Tadqiqotda mavjud ilmiy manbalar asosida mazkur ikki texnologiyaning uyg’unligi, ularning birgalikdagi imkoniyatlari va ta’lim jarayoniga qo’shadigan hissassi ochib berilgan. Taklif etilayotgan integrativ ramka sinov sifati va kod uslubini parallel baholash, ekvivalent mutatsiyalarni aqlli aniqlash hamda talabalar uchun individual o’rganish trayektoriyasini shakllantirishga xizmat qiladi.

Kalit so’zlar: mutatsion testlash, sun’iy intellekt, katta til modellari, kod sifati, avtomatlashtirilgan baholash, raqamli ta’lim, dasturchi tayyorlash

Аннотация : В цифровой образовательной среде при формировании знаний и навыков программирования важное место занимает эффективная оценка качества кода. Традиционная ручная проверка и автоматизированные тесты зачастую недостаточно детальны для выявления индивидуальных ошибок студентов и предоставления обучающей обратной связи. В настоящей статье анализируются инновационные методы оценки качества

кода, основанные на интеграции двух передовых подходов – мутационного тестирования и моделей искусственного интеллекта. Мутационное тестирование измеряет эффективность тестовых наборов с помощью синтетических дефектов (мутантов), тогда как большие языковые модели (SI) способны понимать исходный код и давать персонализированные рекомендации. На основе существующих научных источников в исследовании раскрывается совместимость этих двух технологий, их совместные возможности и вклад в образовательный процесс. Предлагаемая интегративная структура служит для параллельной оценки качества тестирования и стиля кодирования, интеллектуального обнаружения эквивалентных мутантов, а также для формирования индивидуальной траектории обучения студентов.

Ключевые слова: *мутационное тестирование, искусственный интеллект, большие языковые модели, качество кода, автоматизированная оценка, цифровое образование, подготовка программистов*

Abstract: *In digital learning environments, effective evaluation of code quality plays a crucial role in developing programming knowledge and skills. Traditional manual checking and automated tests are often insufficiently detailed for identifying students' individual errors and providing educational feedback. This paper analyzes innovative methods for code quality assessment based on the integration of two advanced approaches—mutation testing and artificial intelligence models. Mutation testing measures the effectiveness of test suites through synthetic defects (mutatsiyas), while large language models (SIs) can understand source code and provide personalized recommendations. Drawing on existing scientific sources, the study reveals the compatibility of these two technologies, their combined capabilities, and their contribution to the educational process. The proposed integrative framework serves to parallelly assess testing quality and coding style, intelligently detect equivalent mutatsiyas, and shape an individual learning*

Keywords: *mutation testing, artificial intelligence, large language models, code quality, automated assessment, digital education, programmer training*

Zamonaviy raqamli ta’lim platformalarida dasturlashni o’rgatish ko’lami kengayib bormoqda. Code.org kabi ommaviy onlayn kurslar millionlab foydalanuvchilarni qamrab olgan bo’lsa, universitetlarning boshlang’ich dasturlash kurslarida ham talabalar soni minglab o’lchanadi [1]. Bunday sharoitda o’qituvchilarning har bir talabaning kodini sifatli baholashi va unga mos individual fikr bildirishi amalda imkonsiz bo’lib qoladi. Avtomatlashtirilgan test tizimlari odatda faqat funksional to’g’rilikni tekshiradi, ammo kodning uslubi, samaradorligi, murakkab mantiqiy xatolari va test qamrovining yetarliligi kabi jihatlarni baholay olmaydi.

Shu sababli, oxirgi yillarda ikki ilg’or yondashuv – mutatsion testlash (mutation testing) va sun’iy intellekt (SI) modellari, xususan katta til modellari (LLM) asosida kod sifatini chuqur baholash usullari jadal rivojlanmoqda. Mutatsion testlash dasturga sun’iy nuqsonlar (mutantlar) kiritib, mavjud test to’plamining ularni topish qobiliyatini o’lchaydi. Bu sinovlarning haqiqiy xatolarni aniqlash salohiyatini bilvosita baholash imkonini beradi [4]. Boshqa tomondan, GPT-4 kabi zamonaviy LLMlar manba kodini semantik tahlil qila oladi, xatolarni lokalizatsiya qiladi, kod uslubi haqida mulohaza bildiradi va hatto tuzatish bo’yicha tavsiyalar yaratadi [1].

Ushbu maqolaning maqsadi – raqamli ta’lim muhitida bo’lajak dasturchi muhandislarni tayyorlashda kod sifatini baholash uchun mutatsion testlash va SI modellarining integratsiyasi imkoniyatlarini ochib berish va ilmiy asosda umumlashtirish. Maqolada so’nggi yillardagi tadqiqot natijalariga tayanib, mazkur ikki texnologiyaning alohida va birgalikdagi afzalliklari, mavjud cheklavlari hamda amaliyotga tatbiq etish yo’llari ko’rib chiqiladi.

Mutatsion testlash ilk bor 1970-yillarda taklif qilingan bo’lib, uning asosida

“kompetent dasturchi gipotezasi” va “bog‘lanish effekti” yotadi [4]. Bu gipotezaga ko‘ra, malakali dasturchilar deyarli to‘g‘ri, kichik sintaktik o‘zgarishlar bilan tuzatilishi mumkin bo‘lgan kod yozadi. Shuning uchun dasturga kiritilgan oddiy sintaktik mutantlar haqiqiy nuqsonlarning vakili bo‘la oladi. Sinov to‘plami mutantlarni “o‘ldirsa” (ya’ni, mutant va asl dastur natijalari orasida farq paydo qilsa), u haqiqiy xatolarni ham aniqlash ehtimoli yuqori bo‘ladi.

Mutatsion testlashning asosiy elementlari:

– Mutatsion operatorlar: sintaktik o‘zgartirish qoidalari (masalan, arifmetik operatorni almashtirish, shartni inkor qilish). Fortran uchun 22 ta, C uchun 77 ta operator to‘plami mavjud [4].

– Mutatsion ball (mutation score): o‘ldirilgan mutantlar sonining jami noekvivalent mutantlarga nisbati. Yuqori ball test to‘plamining ishonchliligini ko‘rsatadi.

– Ekvivalent mutant muammosi: ayrim mutantlar sintaktik farqiga qaramasdan asl dastur bilan bir xil ishlaydi; ularni avtomatik aniqlash hal qilib bo‘lmaydigan muammodir.

Hisoblash xarajatlarini kamaytirish uchun “zaif mutatsiya” (faqat dastur ichki holatini solishtirish), “tanlangan mutatsiya” (eng samarali operatorlarni ishlatish), “mutant sxemasi” (barcha mutantlarni bir metadasturga jamlash) kabi optimallashtirishlar ishlab chiqilgan. So‘nggi ishlarda esa katta hajmli sanoat kodlari va real nuqsonlar bilan solishtirish orqali mutatsion testlashning ishonchliligi tasdiqlangan [7].

So‘nggi o‘n yil ichida dasturlash ta’limida avtomatlashtirilgan baholash tizimlari jadal rivojlandi. Keuning va boshqalar (2018) tomonidan 101 ta vosita tahlil qilinib, aksariyati xatolarni ko‘rsatish bilan cheklangani, lekin o‘quvchiga keyingi qadamni ko‘rsatish bo‘yicha zaif ekanligi aniqlangan [5]. Ushbu bo‘shliqni to‘ldirishda SI texnologiyalari, ayniqsa, katta til modellari katta imkoniyatlar ochdi.

BERT va GPT avlodidagi modellar manba kodini tabiiy tilga o‘xshatib tahlil

qiladi va quyidagi vazifalarni bajara oladi:

– Avtomatik ball qo’yish: GPT-4 o’qituvchi baholari bilan yuqori korrelyatsiyaga ega ekanligi, masalan, Pearson koeffitsienti 0,91 gacha yetganligi qayd etilgan [2].

– Shaxsiylashtirilgan feedback yaratish: GPT-4 nafaqat xatoni ko’rsatadi, balki uni qanday tuzatish, kod uslubini yaxshilash va samaradorlikni oshirish bo’yicha tavsiyalar beradi [1].

– Murakkab tushunchalarni tushuntirish: LLM yordamida kodning qanday ishlashi, qanday algoritm ishlatilgani haqida bosqichma-bosqich izoh berish mumkin.

– Xatolarni lokalizatsiya qilish va avtomatik tuzatish: GPT-4 ko’pincha xato qatorlarni to’g’ri ko’rsatib, tuzatilgan kod variantini taklif etadi [1].

Biroq, ushbu modellarda izchillik, tarafkashlik, gallyutsinatsiya (mavjud bo’lmagan xatolarni ko’rsatish) va maxfiylik kabi jiddiy muammolar mavjud [3]. Shuning uchun inson nazorati va pedagogik integratsiya hal qiluvchi omillar bo’lib qoladi.

Ikki yo’nalishning sintezi bo’yicha ilk qadamlar allaqachon qo’yilgan. Piech va uning hamkasblari (2015) dasturlarni neyron tarmoq orqali chiziqli akslantirish sifatida kodlab, o’qituvchi sharhlarini ommaviy tarqatishda qo’llagan [8]. Bu yondashuv dastur sintaksisini emas, balki uning funksional xususiyatlarini o’rganadi, bu esa mutatsion testlashning asl maqsadi – funksional farqlarni aniqlash bilan uyg’undir. Boshqa tadqiqotlarda LLMlar ekvivalent mutantlarni aniqlash yoki mutant sinovlarini avtomatik generatsiya qilishda sinovdan o’tkazilgan. Masalan, GPT-4 dasturdagi kichik o’zgarishlarni aniqlab, ularning dastur natijasiga ta’sirini talqin qilishi mumkin, bu esa an’anaviy mutatsion testlash tizimlariga sezilarli yordam beradi.

Yuqoridagi tahlillar asosida raqamli ta’lim muhiti uchun quyidagi komponentlarni o’z ichiga olgan integrativ baholash modelini taklif etamiz:

Test sinovlari va mutatsion tahlil bloki:

Talaba kodi avtomatik ravishda oldindan tayyorlangan test to'plami orqali tekshiriladi. Keyin tanlangan mutatsion operatorlar (masalan, asosiy 5 operator) yordamida mutantlar yaratiladi va testlarning “o'ldirish” ko'rsatkichi hisoblanadi. Bu bosqich test qamrovining yetarliligini miqdoriy baholaydi va qaysi kod bloklari sinovdan o'tkazilmaganligini ko'rsatadi [4].

Sun'iy intellektga asoslangan sifat tahlili bloki:

– GPT-4 yoki mahalliy LLM manba kodini to'liq tahlil qilib, sintaktik xatolar, mantiqiy kamchiliklar, kod uslubi buzilishlari va hujjatlashtirish darajasini baholaydi [1].

– Model natijalariga asoslanib, talabaga mos ravishda shaxsiylashtirilgan matnli tavsiyalar generatsiya qilinadi: xatolarning izohi, to'g'rilash bo'yicha takliflar, kod namunasi yoki navbatdagi qadam uchun yo'riqnoma.

– Mutatsion tahlil natijalari bilan birgalikda, LLM test qamrovi past bo'lgan qismlar uchun qo'shimcha test yozish bo'yicha ham maslahat beradi.

Ekvivalent mutantlarni aqlli filtrlash:

An'anaviy mutatsion testlashning eng katta qiyinchiligi – ekvivalent mutantlarni aniqlash. LLM yordamida ikki kod fragmentining semantik ekvivalentligini taxmin qilish va shubhali mutantlarni o'qituvchi yoki talaba ko'rib chiqishi uchun ajratib berish mumkin [7]. Bu qo'lda sarflanadigan vaqtni keskin kamaytiradi.

Orkestrator va interfeys:

LMS (Moodle va hokazo) bilan integratsiyalashgan modul barcha baholash natijalarini yig'adi, talabaga tushunarli hisobot shaklida taqdim etadi va o'qituvchiga umumiy sinf statistikasi, qiyin mavzular va xavf guruhidagi talabalar haqida ma'lumot beradi [6].

Taklif etilayotgan ramkaning afzalligi shundaki, u faqat funksional to'g'rilikni emas, balki test sifatini, kodning o'qiluvchanligini, ishlash unumdorligini va

talabaning taraqqiyotini bir vaqtning o‘zida baholaydi.

Tahlil shuni ko‘rsatadiki, mutatsion testlash va SI modellari bir-birini to‘ldiradi: mutatsion testlash test to‘plamining “chuqurligini” o‘lchasa, LLMlar kodning “kengligi” – uslubi, hujjatlashtirilishi, murakkabligi va xatolarning ma’nosini tushuntiradi. Integratsiya, ayniqsa, katta hajmdagi dasturlash kurslarida o‘qituvchining vaqtini tejaydi, talabalar uchun darhol va mazmunli fikr olish imkonini yaratadi.

Biroq, amaliyotga tatbiq etishda quyidagi cheklovlar va muammolar e’tiborga olinishi lozim:

- Hisoblash xarajatlari: mutantlarni yaratish va LLM so‘rovlari katta resurs talab qilishi mumkin. Optimallashtirilgan tanlangan mutatsiya va mahalliy (on-premise) kichik LLMlar bu muammoni yengillashtiradi.

- Pedagogik moslik: avtomatik feedback talabalarning motivatsiyasi va mustaqil fikrlashiga salbiy ta’sir ko‘rsatmasligi uchun ehtiyotkorlik bilan loyihalanishi kerak.

- Maxfiylik va axloqiy masalalar: talabalar kodini tashqi SI xizmatlariga yuborish maxfiylik xavfini tug‘diradi. Ochiq kodli va mahalliy serverlarda ishlovchi yechimlar tavsiya etiladi.

- Inson nazorati: yakuniy baholashda o‘qituvchining roli saqlanib qolishi, SI faqat yordamchi vosita sifatida xizmat qilishi zarur.

Kelgusidagi tadqiqotlar real o‘quv sharoitida ushbu integrativ yondashuvning samaradorligini empirik baholash, LLM yordamida yuqori tartibli mutantlarni generatsiya qilish va talabalarning individual o‘rganish stillariga mos moslashuvchan tizimlarni yaratishga qaratilishi mumkin [4].

Raqamli ta’lim muhitida bo‘lajak dasturchi muhandislarni tayyorlashda kod sifatini baholashning an’anaviy usullari endilikda mutatsion testlash va sun’iy intellekt modellarining integratsiyasi bilan yangi bosqichga ko‘tarilmoqda. Mutatsion testlash test to‘plamlarining ishonchliligini chuqur tekshirish imkonini

bersa, LLMlar kodni semantik tushunib, shaxsiylashtirilgan va ko‘p qirrali teskari aloqa yaratadi. Ushbu maqolada taklif etilgan integrativ ramka ikki texnologiyaning kuchli tomonlarini birlashtirib, o‘qituvchilar uchun vaqtni tejash va talabalar uchun individual rivojlanishni qo‘llab-quvvatlashga qaratilgan. Amaliy tatbiq yo‘lidagi cheklovlar bartaraf etilsa, mazkur yondashuv ommaviy dasturlash ta’limida sifat va qamrovni yangi darajaga olib chiqishga qodir.

FOYDALNILGAN ADABIYOTLAR RO‘YXATI

1. Azazi, I., Kiesler, N., & Strickroth, S. Feedback-Generation for Programming Exercises With GPT-4. Proceedings of ITiCSE 2024.
2. Bernik, A., Radošević, D., & Cep, A. A Comparative Study of Large Language Models in Programming Education: Accuracy, Efficiency, and Feedback. Applied Sciences, 15, 10055. 2025.
3. Emirtekin, E. Large Language Model-Powered Automated Assessment: A Systematic Review. Applied Sciences, 15, 5683. 2025.
4. Jia, Y., & Harman, M. An Analysis and Survey of the Development of Mutation Testing. IEEE Transactions on Software Engineering, 37(5), 649–678. 2011.
5. Keuning, H., Jeuring, J., & Heeren, B. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. ACM Transactions on Computing Education, 19(1), Article 3. 2018.
6. Manorat, P., Tuarob, S., & Pongpaichet, S. Artificial intelligence in computer programming education: A systematic literature review. Computers and Education: Artificial Intelligence, 8, 100403. 2025.
7. Papadakis, M., Kintis, M., Zhang, J., Jia, Y., Le Traon, Y., & Harman, M. Mutation Testing Advances: An Analysis and Survey. In Advances in Computers (Vol. 112, pp. 275–378). Elsevier. 2019.
8. Piech, C., Huang, J., Nguyen, A., Phulsuksombati, M., Sahami, M., & Guibas, L. Learning Program Embeddings to Propagate Feedback on Student Code. International Conference on Machine Learning. 2015.

