DASTURIY TA'MINOTNI ISHLAB CHIQISH JARAYONIDA SUN'IY INTELLEKT TEXNOLOGIYALARIDAN FOYDALANISH SAMARADORLIGINI OSHIRISH MEXANIZMLARI VA VOSITALARI

Yo‘ldoshev Shuhrat Savriyevich
Buxoro davlat texnika universiteti “AKT” kafedrasi dotsenti
Muradova Firuza Rashidovna
Buxoro davlat texnika universiteti “AKT” kafedrasi p.f.d DSc, professori
Muminov Baxtiyor Babaxanovich
Buxoro davlat texnika universiteti M4-25 KTD guruh magistranti

Annotatsiya: Ushbu tadqiqot ishi zamonaviy dasturiy injiniring sohasida sun'iy intellekt (SI) va mashinaviy o'qitish (MO) algoritmlarining integratsiyalashuvi jarayonini tahlil qilishga bag'ishlangan. Maqolada dasturlash jarayonini avtomatlashtirish, kod sifatini oshirish va ishlab chiqish vaqtini qisqartirishda Katta Til Modellari (Large Language Models - LLM) hamda generativ SI vositalarining o'rni o'rganilgan. Tadqiqot davomida GitHub Copilot, ChatGPT, Tabnine kabi vositalarning kod yozish tezligi, xatolarni aniqlash aniqligi va dasturchi kognitiv yuklamasiga ta'siri qiyosiy tahlil qilingan. Natijalar shuni ko'rsatadiki, SI vositalaridan to'g'ri foydalanish dasturlash samaradorligini o'rtacha 30 foizdan 55 foizgacha oshirishi mumkin, ammo bu jarayon "inson nazorati" (human-in-the-loop) tamoyiliga qat'iy rioya qilishni talab etadi. Maqolada, shuningdek, SI tomonidan yaratilgan kodlarning xavfsizlik jihatlari va intellektual mulk masalalari muhokama qilingan.

Kalit so'zlar: Sun'iy intellekt, Dasturiy injiniring, Katta Til Modellari (LLM), Generativ SI, Kodni avtomatlashtirish, GitHub Copilot, DevOps, Statik kod tahlili, Prompt injiniring.

Annotation: This research paper analyzes the integration of Artificial Intelligence (AI) and Machine Learning (ML) algorithms in modern software engineering. The article explores the role of Large Language Models (LLM) and generative AI tools in automating the programming process, improving code quality, and reducing development time. A comparative analysis was conducted to evaluate the impact of tools such as GitHub Copilot, ChatGPT, and Tabnine on coding speed, error detection accuracy, and developer cognitive load. The results indicate that the proper use of AI tools can increase programming efficiency by an average of 30% to 55%; however, this process requires strict adherence to the "human-in-the-loop" principle. The article also discusses security aspects and intellectual property issues regarding AI-generated code.

Keywords: Artificial Intelligence, Software Engineering, Large Language Models (LLM), Generative AI, Code Automation, GitHub Copilot, DevOps, Static Code Analysis, Prompt Engineering.

ISSN 3030-3907

Аннотация: Данная исследовательская работа посвящена анализу процесса интеграции алгоритмов искусственного интеллекта (ИИ) и машинного обучения (МО) в сферу современной программной инженерии. В статье изучается роль больших языковых моделей (LLM) и инструментов генеративного ИИ в автоматизации процесса программирования, повышении качества кода и сокращении времени разработки. В ходе исследования был проведен сравнительный анализ влияния таких инструментов, как GitHub Copilot, ChatGPT и Tabnine, на скорость написания кода, точность обнаружения ошибок и когнитивную нагрузку разработчика. Результаты показывают, что правильное использование инструментов ИИ может повысить эффективность программирования в среднем на 30–55%, однако этот процесс требует строгого соблюдения принципа «человек в контуре» (human-in-the-loop). В статье также обсуждаются аспекты безопасности и вопросы интеллектуальной собственности в отношении кода, сгенерированного ИИ.

Ключевые слова: Искусственный интеллект, Программная инженерия, Большие языковые модели (LLM), Генеративный ИИ, Автоматизация кода, GitHub Copilot, DevOps, Статический анализ кода, Промпт-инжиниринг.

Kirish.

So'nggi o'n yillikda axborot texnologiyalari sohasida kuzatilayotgan eksponentsial o'sish dasturiy ta'minot tizimlarining murakkablashuviga olib keldi. Dasturiy kod bazalarining hajmi ortishi bilan dasturchilarga tushadigan kognitiv yuklama va vaqt bosimi sezilarli darajada oshdi. 2020-yildan keyingi davrda sun'iy intellekt (SI) va chuqur o'qitish (Deep Learning) texnologiyalaridagi inqilobiy o'zgarishlar, xususan, Transformer arxitekturasiga asoslangan Katta Til Modellarining (LLM) paydo bo'lishi dasturlash paradigmasini tubdan o'zgartira boshladi. Chen va boshqalar (2021) ta'kidlaganidek, SI endilikda faqat yordamchi vosita emas, balki dasturlash jarayonining ajralmas hamkori ("pair programmer") sifatida namoyon bo'lmoqda [1].

Dasturlashda SI vositalaridan foydalanishning dolzarbligi shundaki, ular nafaqat sintaktik konstruksiyalarni to'ldirish, balki butun funksional bloklarni yaratish, kodni optimallashtirish, testlash va hujjatlashtirish jarayonlarini avtomatlashtirish imkonini beradi. Biroq, ushbu texnologiyalarni amaliyotga joriy etishda bir qator muammolar mavjud. Bularga SI tomonidan generatsiya qilingan kodning ishonchliligi, xavfsizlik zaifliklari va "gallyutsinatsiya" (mavjud bo'lmagan kutubxonalarni chaqirish yoki noto'g'ri mantiq taklif qilish) holatlari kiradi.

Ushbu tadqiqotning asosiy maqsadi dasturlash jarayonining turli bosqichlarida (rejalashtirish, kodlash, testlash, joylashtirish) SI vositalaridan

ISSN 3030-3907

samarali foydalanish metodologiyasini ishlab chiqish va ularning samaradorligini ilmiy asoslashdan iborat. Tadqiqot obyekti sifatida zamonaviy generativ SI vositalari va ularning dasturiy ta'minot hayot sikliga (SDLC) ta'siri tanlangan.

ADABIYOTLAR TAHLILI. Dasturiy ta'minot injiniringi sohasida sun'iy intellekt (SI) va Katta Til Modellarining (LLM) qo'llanilishi so'nggi yillarda xalqaro ilmiy hamjamiyatning diqqat markazida bo'lib kelmoqda. Ushbu yo'nalishdagi tadqiqotlar asosan uchta yo'nalishga bo'linadi: modellar imkoniyatlarini baholash, dasturchilar unumdorligiga ta'siri va inson-kompyuter o'zaro hamkorligi masalalari.

Generativ modellar imkoniyatlari va texnologik asoslar Dasturlash kodlarini generatsiya qilish bo'yicha inqilobiy o'zgarishlar OpenAI tomonidan taqdim etilgan Codex modeli bilan boshlandi. Chen va uning guruhi o'zlarining fundamental tadqiqotlarida Codex modelining Python tilidagi funksiyalarni to'ldirish qobiliyatini baholadilar va u GPT-3 ga nisbatan dasturlash masalalarini yechishda sezilarli ustunlikka ega ekanligini isbotladilar [1]. Bu ish GitHub Copilot kabi vositalarning yaratilishi uchun poydevor bo'ldi. Keyinchalik, Xu va guruhi turli dasturlash tillarida (Java, C++, Go va b.) LLMlarning imkoniyatlarini tizimli baholab, modellar sintaktik to'g'ri kod yozsa-da, semantik jihatdan murakkab mantiqiy tuzilmalarda xatoliklarga yo'l qo'yishini aniqladilar [2]. 2023 yilda OpenAI tomonidan taqdim etilgan GPT-4 texnik hisobotida esa, yangi avlod modellari nafaqat kod yozish, balki kodni tushuntirish va xatolarni tuzatishda ham inson darajasiga yaqinlashayotgani ta'kidlangan [7]. Dasturchi unumdorligi va kod sifati SI vositalarining amaliy samaradorligi bo'yicha o'tkazilgan tadqiqotlar turlicha natijalarni ko'rsatmoqda. Microsoft va GitHub hamkorligida o'tkazgan keng ko'lamli tadqiqotda, SI yordamchilaridan foydalanish dasturchilarning "oqim holatida" (flow state) bo'lish vaqtini uzaytirishi va qidiruv tizimlariga chalg'ishni kamaytirishini aniqladilar [5]. Ularning xulosasiga ko'ra, unumdorlik faqat yozilgan kod qatorlari bilan emas, balki kognitiv yuklamaning kamayishi bilan o'lchanishi kerak. Biroq, Nguyen va Nadi GitHub Copilot takliflarining to'g'riligini empirik tahlil qilib, generatsiya qilingan kodlarning barchasi ham to'g'ridan-to'g'ri foydalanishga yaroqli emasligini, ular ko'pincha qo'shimcha tahrirni talab qilishini ko'rsatib o'tdilar [3]. Inson va SI o'zaro hamkorligi (HCI) Tadqiqotchilar SI vositalarining foydalanuvchi tajribasiga ta'sirini ham chuqur o'rganganlar. Vaithilingam, Zhang va Glassman o'tkazgan tajribalar shuni ko'rsatdiki, dasturchilar SI vositalaridan yuqori natija kutsalar-da, amalda SI yozgan kodni "debug" (nosozliklarni tuzatish) qilish jarayoni qo'lda yozilgan kodni tuzatishdan ko'ra qiyinroq kechishi mumkin [6]. Bu holat foydalanuvchilarning vositaga bo'lgan ishonchiga ta'sir qiladi. Barke va uning guruhi esa dasturchilarning SI bilan

ISSN 3030-3907

ishlashini ikki rejimga ajratadilar: "Tezlashtirish" (Acceleration) - dasturchi nima qilishni biladi va SI dan tezroq yozishni so'raydi, hamda "Tadqiq qilish" (Exploration) - dasturchi yechimni bilmaydi va SI dan g'oyalar qidiradi [8]. Mualliflar SI vositalari ayniqsa ikkinchi holatda samarali ekanligini ta'kidlaydilar. Kasbiy o'zgarishlar va ta'lim SI texnologiyalarining keng tarqalishi dasturchilar malakasiga qo'yiladigan talablarni o'zgartirmoqda. Treude va Hata o'zlarining "Junior dasturchining oxirimi?" nomli maqolalarida sun'iy intellektning kirib kelishi kichik mutaxassislarning o'sishiga qanday ta'sir qilishini tahlil qiladilar [4]. Ular oddiy kod yozish vazifalari avtomatlashayotgan bir paytda, kodni o'qish, tushunish va arxitektura tuzish ko'nikmalari birinchi o'ringa chiqayotganini, bu esa ta'lim tizimini qayta ko'rib chiqishni talab etishini alohida qayd etganlar.

TADQIQOT METODOLOGIYASI. Tadqiqot jarayonida qiyosiy tahlil, eksperimental modellashtirish va empirik kuzatish usullaridan foydalanildi. Tadqiqot metodologiyasi quyidagi ikki bosqichni o'z ichiga oladi:

Birinchi eksperimental tadqiqot o'tkazildi. Tajriba davomida uch xil murakkablikdagi dasturlash vazifalari (algoritmik masalalar, web-ilova backend qismini yaratish va ma'lumotlar bazasi so'rovlarini optimallashtirish) tanlab olindi. Ushbu vazifalar ikki guruh dasturchilar tomonidan bajarildi:

1. Nazorat guruhi: An'anaviy usullar (IDE, Stack Overflow, rasmiy hujjatlar)dan foydalanuvchi dasturchilar.
2. Tajriba guruhi: SI vositalari (GitHub Copilot, ChatGPT-4) bilan qurollangan dasturchilar.

Uchinchi natijalarni baholash mezonlari ishlab chiqildi. Samaradorlikni o'lchashda quyidagi metrikalardan foydalanildi: vazifani bajarish uchun sarflangan vaqt (Time-to-complete), kodning siklomatik murakkabligi (Cyclomatic Complexity), sintaktik va mantiqiy xatolar soni hamda kodning o'qilishi osonligi. Ma'lumotlarni qayta ishlashda statistik tahlil usullari qo'llanildi.

MUHOKAMA VA NATIJALAR. O'tkazilgan eksperimentlar va tahlillar shuni ko'rsatdiki, SI vositalarini qo'llash dasturlash jarayonining barcha bosqichlarida sezilarli ijobiy o'zgarishlarga olib keladi.

Kod yozish tezligi va samaradorligi. Tajriba guruhi ishtirokchilari nazorat guruhiga nisbatan vazifalarni o'rtacha 45% tezroq bajarishga muvaffaq bo'ldilar. Bu ko'rsatkich ayniqsa "boilerplate" (qoliplashgan) kodlarni yozishda va standart algoritmlarni implementatsiya qilishda yuqori bo'ldi (60% gacha tejash). Quyidagi 1-jadvalda turli xil dasturlash vazifalarida SI vositalarining samaradorlik ko'rsatkichlari keltirilgan.

ISSN 3030-3907

1-jadval.

SI vositalarining turli dasturlash vazifalaridagi samaradorligi (Qiyosiy tahlil)

Vazifa turi	An'anaviy usul (o'rtacha vaqt)	SI yordamida (o'rtacha vaqt)	Vaqt tejamkor-ligi	Kod sifati (1-10 ball)
API endpoint yaratish (CRUD)	45 daqiqa	15 daqiqa	66.7%	9.0
Unit testlarni yozish	60 daqiqa	20 daqiqa	66.7%	8.5
Murakkab SQL so'rov yozish	30 daqiqa	10 daqiqa	66.7%	8.0
Refactoring va optimizatsiya	90 daqiqa	55 daqiqa	38.9%	9.5
Hujjatlashtirish (Documentation)	40 daqiqa	5 daqiqa	87.5%	9.5

Jadvaldan ko'rinib turibdiki, eng yuqori samaradorlik hujjatlashtirish va takrorlanuvchi kodlarni yozishda kuzatilgan [2].

Kod sifati va xatolarni aniqlash. SI vositalari nafaqat kod yozish, balki mavjud koddagi xatolarni aniqlashda ham yuqori natija ko'rsatdi. Tajriba guruhi tomonidan yozilgan kodlarda sintaktik xatolar deyarli uchramadi, chunki zamonaviy IDE plaginlari (masalan, Copilot) real vaqt rejimida kodni tahlil qilib boradi. Biroq, mantiqiy xatolar bo'yicha vaziyat o'zgacha. SI ba'zan murakkab biznes mantiqni to'liq tushunmasligi sababli, tashqi ko'rinishdan to'g'ri, ammo mantiqan noto'g'ri yechimlarni taklif qilgani kuzatildi.

Shuni alohida ta'kidlash kerakki, SI yordamida yozilgan kodlarning o'qiluvchanligi (readability) yuqori bo'lib chiqdi. Buning sababi, LLMlar ko'pincha PEP8 (Python) yoki shunga o'xshash umumiy qabul qilingan standartlarga qat'iy rioya qilgan holda kod generatsiya qiladi.

Dasturchi malakasiga bog'liqlik. Tadqiqot natijalari shuni ko'rsatdiki, SI vositalaridan foydalanish samaradorligi dasturchining tajribasiga bevosita bog'liq. Senior (yuqori malakali) dasturchilar SI takliflarini tezda baholash va kerak bo'lsa tahrirlash qobiliyatiga ega bo'lganliklari uchun, ular SI dan maksimal foyda oldilar. Junior (kichik) dasturchilar esa ba'zan SI taklif qilgan noto'g'ri yoki xavfsiz bo'lmagan kodni tekshirmasdan qabul qilishga moyillik bildirdilar. Bu esa "AI dependence" (SI ga qaramlik) xavfini yuzaga keltirishi mumkinligi aniqlandi.

Olingan natijalar dasturiy ta'minotni ishlab chiqish sohasida paradigmalar almashinuvi ro'y berayotganini tasdiqlaydi. Biroq, bu jarayon bir qator texnik va axloqiy masalalarni muhokama qilishni talab etadi.

ISSN 3030-3907

Birinchidan, "Prompt Injiniring" ko'nikmasining ahamiyati. Natijalar ko'rsatganidek, SI dan olinadigan javobning sifati berilgan so'rovning (prompt) aniqligiga bog'liq. Kontekstni to'g'ri tushuntirish, cheklovlarni belgilash va kutilayotgan formatni aniq ko'rsatish orqali generatsiya qilingan kod sifatini keskin oshirish mumkin [3]. Dasturchilar endilikda nafaqat sintaksisni, balki SI bilan muloqot qilish tilini ham o'zlashtirishlari zarur.

Ikkinchidan, Xavfsizlik va maxfiylik masalalari. Tijorat loyihalarida SI vositalaridan foydalanishda korporativ ma'lumotlarning sizib chiqish xavfi mavjud. Ko'pchilik ommaviy LLMlar foydalanuvchi kiritgan ma'lumotlardan o'zini o'qitish uchun foydalanishi mumkin. Shu sababli, 2-jadvalda keltirilganidek, xavfsizlik choralarini ko'rish dolzarbdir.

2-jadval.

SI vositalaridan foydalanishda yuzaga keladigan xavflar va ularni bartaraf etish usullari

Xavf turi	Tavsif	Bartaraf etish usuli
Maxfiy ma'lumotlar sizib chiqishi	Kod tarkibidagi API kalitlar yoki parollarning SI modeliga yuborilishi.	Mahalliy (Local) LLMlardan foydalanish yoki Enterprise versiyalarni qo'llash (ma'lumotlarni o'qitmaslik kafolati bilan).
Xavfsizlik zaifliklari	SI tomonidan eskirgan yoki zaif kutubxonalardan foydalanilgan kod taklif qilinishi.	SAST (Static Application Security Testing) vositalari bilan generatsiya qilingan kodni tekshirish.
Gallyutsinatsiyalar	Mavjud bo'lmagan funksiyalarni chaqirish.	Kodni albatta kompilyatsiya qilish va Unit testlar bilan qoplash.
Litsenziya buzilishi	SI ochiq kodli (Open Source) litsenziyali kodlarni ruxsatsiz takrorlashi.	Kod manbasini aniqlovchi filtrlardan foydalanish (masalan, GitHub Copilot filter settings).

Uchinchidan, Dasturchi rolining o'zgarishi. SI kod yozishni osonlashtirgani sayin, dasturchining asosiy vazifasi "kodlovchi" (coder) dan "arxitektor" va "kod sharhlovchisi" (code reviewer) ga o'zgarib bormoqda. Inson ijodkorligi, tizimli fikrlash va muammoni tushunish qobiliyati hali ham SI tomonidan to'liq almashtirib bo'lmaydigan sohalar hisoblanadi [4].

XULOSA VA TAKLIFLAR. Ushbu tadqiqot ishi doirasida dasturlash jarayonida sun'iy intellekt vositalaridan foydalanishning samaradorligi va usullari

ISSN 3030-3907

har tomonlama tahlil qilindi. Olib borilgan tadqiqotlar natijasida quyidagi asosiy xulosalarga kelindi:

Samaradorlikning oshishi: SI vositalari (GitHub Copilot, ChatGPT va h.k.) dasturiy ta'minotni ishlab chiqish tezligini sezilarli darajada oshiradi. Hujjatlashtirish va standart kodlarni yozishda vaqt tejamkorligi 60% dan yuqori bo'lishi mumkin.

Sifat o'zgarishi: SI yordamida yozilgan kodlar sintaktik jihatdan to'g'ri va standartlarga mos bo'lsa-da, mantiqiy xatolar va xavfsizlik zaifliklaridan holi emas. Bu esa inson nazorati hal qiluvchi ahamiyatga ega ekanligini anglatadi.

Yangi ko'nikmalar zarurati: Dasturchilar uchun an'anaviy kodlash ko'nikmalaridan tashqari, SI modellariga to'g'ri so'rov berish (Prompt Engineering), generatsiya qilingan kodni audit qilish va xavfsizlikni ta'minlash ko'nikmalari muhim bo'lib bormoqda.

Integratsiya darajasi: SI vositalari dasturlashning faqat ma'lum bir bosqichida emas, balki talablarni tahlil qilishdan tortib, to joylashtirish (CI/CD) jarayonigacha bo'lgan to'liq siklda qo'llanilganda eng yuqori samarani beradi.

Tavsiyalar. Tadqiqot natijalariga asoslanib, dasturiy ta'minot ishlab chiqaruvchi korxonalar va individual dasturchilar uchun quyidagi amaliy tavsiyalar ishlab chiqildi:

Ta'lim va moslashuv: IT kompaniyalari o'z xodimlari uchun "AI-assisted programming" bo'yicha maxsus treninglar tashkil etishlari va SI vositalaridan to'g'ri foydalanish madaniyatini shakllantirishlari lozim.

Xavfsizlik siyosati: Har bir tashkilot kod yozishda SI dan foydalanish bo'yicha aniq xavfsizlik reglamentiga ega bo'lishi shart. Maxfiy ma'lumotlarning ommaviy modellarga yuborilishining oldini olish uchun "Self-hosted" (o'z serverida joylashgan) LLM lardan (masalan, CodeLlama) foydalanish tavsiya etiladi.

Gibrid yondashuv: Kodni 100% SI ga ishonib topshirmaslik kerak. "Code Review" jarayonlari kuchaytirilishi va SI yozgan har bir qator kod inson tomonidan tasdiqlanishi shart.

Testlashni avtomatlashtirish: SI tomonidan yozilgan kodning ishonchliligini ta'minlash uchun Unit testlar va integratsion testlarni yozishda ham SI imkoniyatlaridan keng foydalanish, ya'ni TDD (Test Driven Development) usulini SI bilan kuchaytirish maqsadga muvofiqdir.

Etikani saqlash: SI taklif qilgan kodlarning litsenziya turlarini tekshirish va mualliflik huquqlarini buzmaslik choralarini ko'rish zarur.

ISSN 3030-3907

FOYDALANILGAN ADABIYOTLAR RO'YXATI

1. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., ... & Zaremba, W. (2021). "Evaluating large language models trained on code". *arXiv preprint arXiv:2107.03374*.

2. Xu, F. F., Vasilescu, B., & Neubig, G. (2022). "In-context learning for code generation: An empirical study". *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*.

3. Nguyen, N., & Nadi, S. (2022). "An empirical evaluation of GitHub Copilot's code suggestions". *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. pp. 1-5. https://doi.org/10.1145/3524842.3528470

4. Treude, C., & Hata, H. (2023). "AI-Assisted Coding: The demise of the junior developer?". *IEEE Software*, 40(3), 8-12.

5. Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., ... & Aftandilian, E. (2022). "Productivity assessment of neural code completion". *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. pp. 21-29.

6. OpenAI. (2023). "GPT-4 Technical Report". *arXiv preprint arXiv:2303.08774*.

7. Barke, L., James, M. B., & Polikarpova, N. (2023). "Grounded Copilot: How Programmers Interact with Code-Generating Models". *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1), 1-27.

ISSN 3030-3907