

**C# DASTURLARIDA XATOLARNI ANIQLASHDA
KATTA TIL MODELLARI VA STATIK TAHLILNI
BIRLASHTIRISH YONDASHUVI**

Muallif: **Qurtov M.X.** | Tashkilot: uzinfocom

E-pochta: kurtov.mansur@gmail.com

Konferensiya maqolasi • Metodologik taklif

Annotatsiya: *Dasturiy ta'minotdagi xatolarni erta aniqlash uning ishonchliligi va xavfsizligini oshirishda muhim ahamiyatga ega. Statik tahlil vositalari kodni oldindan belgilangan qoidalar asosida tekshiradi, katta til modellari (LLM) esa kod kontekstini sharhlash va ehtimoliy muammolarni tabiiy tilda tushuntirish imkonini beradi. Biroq har ikki yondashuvda ham cheklovlar mavjud: statik analizatorlar ayrim kontekstga bog'liq xatolarni o'tkazib yuborishi, LLM esa mavjud bo'lmagan xato haqida ishonarli ko'rinadigan xabar yaratishi mumkin. Ushbu maqolada C# dasturlaridagi xatolarni aniqlash uchun LLM va statik tahlil natijalarini birlashtiruvchi metodologik yondashuv taklif etiladi. Yondashuv uchta variantni — faqat LLM, faqat statik tahlil va ularning birlashtirilgan variantini — bir xil etalon ma'lumotlar to'plamida baholashni nazarda tutadi. Baholash uchun precision, recall, F1-score, noto'g'ri ijobiy topilmalar ulushi, ishlash vaqti va xarajat ko'rsatkichlari tavsiya qilinadi. Maqola tajriba dizayni, ma'lumotlarni tanlash, topilmalarni moslashtirish va takrorlanuvchanlik talablarini bayon qiladi. Bu ish metodologik taklif bo'lib, unda hali bajarilmagan tajribalarning natijalari keltirilmaydi; yondashuvning samaradorligi keyingi empirik tekshiruv orqali aniqlanishi kerak.*

Kalit so'zlar: *sun'iy intellekt, katta til modeli, C#, dasturiy xatolar, statik tahlil, CodeQL, dasturiy ta'minot sifati, F1-score.*

Abstract: *Early detection of software defects is important for improving software reliability and security. Static-analysis tools inspect source code against predefined rules, while large language models (LLMs) can interpret code context and explain potential problems in natural language. Both approaches have limitations: static analyzers may miss context-dependent defects, whereas LLMs may produce plausible but unsupported findings. This paper proposes a methodological approach that combines LLM findings with static analysis for defect detection in C# applications. The approach compares three configurations on the same labeled dataset: LLM-only, static-analysis-only, and a combined workflow. Precision, recall, F1-score, false-positive rate, execution time, and cost are recommended as evaluation measures. The paper describes dataset selection, finding-matching rules, experimental design, and reproducibility requirements. It is a methodological proposal; no results from experiments that have not been conducted are reported. The effectiveness of the approach must be established through a subsequent empirical evaluation.*

Keywords: *artificial intelligence, large language model, C#, software defects, static analysis, CodeQL, software quality, F1-score.*

1. Kirish

Dasturiy ta'minot ishlab chiqish jarayonida xatolarni imkon qadar erta aniqlash tuzatish xarajatlarini kamaytirishi, tizim ishonchliligini oshirishi va xavfsizlik bilan bog'liq xavflarni pasaytirishi mumkin. Statik tahlil vositalari dastur bajarilmasdan turib manba kodini tekshiradi va belgilangan qoidalar bo'yicha ogohlantirishlar chiqaradi. CodeQL kabi vositalar kodni so'rovlar orqali tahlil qilish va ma'lum muammoli naqshlarni aniqlash imkonini beradi [1].

Katta til modellari kodni izohlash, ehtimoliy xatolarni ko'rsatish va tuzatish variantlarini taklif qilishda qo'shimcha imkoniyat yaratadi. Shunga qaramay, LLM yaratgan har bir topilma to'g'ri deb qabul qilinmasligi kerak:

model talablar bilan tasdiqlanmagan xatoni taxmin qilishi yoki noto'g'ri tuzatish taklif qilishi mumkin. Shu sababli LLM natijalarini mustaqil tekshirish va an'anaviy vositalar bilan solishtirish zarur.

Ushbu maqolaning maqsadi C# dasturlaridagi xatolarni aniqlashda LLM va statik tahlilni birlashtirish uchun takrorlanadigan metodologiyani taklif etishdir. Tadqiqot quyidagi savollarga yo'naltirilgan: (1) LLM etalon xatolarni qay darajada aniqlaydi? (2) Statik tahlil bilan aniqlangan xatolar qanchalik mos keladi yoki farqlanadi? (3) Natijalarni birlashtirish noto'g'ri topilmalarni nazorat qilgan holda qamrovni oshirishi mumkinmi?

Maqolaning amaliy hissasi — uchta yondashuvni bir xil ma'lumotlar va bir xil baholash qoidalari asosida tekshirish rejasi, topilmalarni moslashtirish mezonlari va takrorlanuvchanlik bo'yicha tavsiyalardir. Ushbu maqola tajriba natijalari e'lon qilinayotgan empirik ish emas, balki empirik tekshiruv uchun metodologik taklif sifatida taqdim etiladi.

2. Mavjud ishlar va nazariy asos

Statik tahlil vositalari ko'pincha aniq qoidalar va ma'lum xato naqshlariga tayangan holda natija beradi. Buning afzalligi — natijalarni qayta olish va qoidalar asosida izohlash imkoniyatidir. Biroq qoida to'plami qamrab olmagan holatlar yoki ilovaning biznes-mantiqiga bog'liq xatolar aniqlanmay qolishi mumkin [1].

LLMlarni dasturiy ta'minot muhandisligi vazifalarida baholash bo'yicha tadqiqotlar aniq vazifa, etalon natija va qayta bajariladigan tajriba zarurligini ko'rsatadi. SWE-bench, masalan, haqiqiy GitHub muammolarini hal qilishga qaratilgan bo'lib, model yaratgan kod o'zgarishlarini real loyiha kontekstida tekshiradi [2]. Ushbu maqoladagi vazifa torroq: u C# kodida mavjud xatolarni topish va topilmalarni statik tahlil natijalari bilan solishtirishga qaratilgan.

LLM va statik tahlilni birlashtirish g'oyasi ularning bir-birini to'ldirishi mumkinligiga asoslanadi. Biroq bu taxminni natija sifatida ko'rsatib bo'lmaydi: birlashtirish noto'g'ri ijobiy xabarlarni ham ko'paytirishi mumkin.

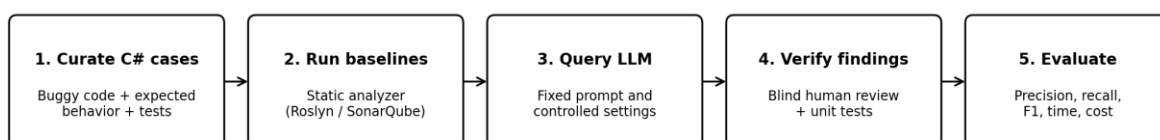
Shu sababli kombinatsiyalangan usul alohida, oldindan belgilangan mezonlar bilan tekshirilishi lozim.

3. Taklif etilayotgan metodologiya

3.1. Tadqiqot dizayni

Tadqiqotda bir xil C# kod namunalari uchta konfiguratsiyada tekshiriladi: A — faqat LLM; B — faqat statik analizator; C — LLM va statik analizator natijalarini birlashtirish. Taqqoslash adolatli bo‘lishi uchun barcha usullar bir xil yakuniy test to‘plamida ishlatiladi. So‘rov matni, model identifikatori, analizator versiyasi va qoidalar to‘plami yakuniy tajribadan oldin qayd etiladi.

Proposed reproducible evaluation workflow



Important: this diagram describes the planned method; it does not represent completed experiments.

1-rasm. Tajribani tashkil etish uchun tavsiya etiladigan ish jarayoni.

3.2. Ma'lumotlar to'plami

Ochiq ma'lumotlar manbasi sifatida Microsoft'ning InferredBugs repozitoriysi dastlabki nomzod sifatida ko'rib chiqilishi mumkin [3]. Unda xatolar va ularni tuzatish bilan bog'liq ma'lumotlar mavjud. Repozitoriydan foydalanishdan oldin C# namunalari, metama'lumotlar, litsenziya va etalon xatolarni qayta tekshirish imkoniyati alohida tekshirilishi kerak. To'plam avvalgi analizator natijalariga asoslangan bo'lsa, shu holat tadqiqot cheklovi sifatida qayd etiladi.

Tanlangan misollarda xato turi, xato joyi, tuzatilishidan oldingi kod va tuzatilgan kod yoki test dalili mavjud bo'lishi maqsadga muvofiq. Deyarli bir xil misollar takrorlanmasligi va bir xil kodning o'xshash nusxalari ishlab

chiqish hamda yakuniy test to'plamlariga bir vaqtda tushmasligi kerak. Yakuniy test to'plami so'rovni sozlash uchun ishlatilmasligi lozim.

3.3. LLM va statik analizator sozlamalari

LLM uchun aniq model identifikatori, provayder, foydalanish sanasi, parametrlar va so'rov matni qayd etiladi. Tavsiya etiladigan so'rov modeldan faqat konkret, dalil bilan izohlanadigan xatolarni qaytarishni so'raydi; uslubiy afzalliklar xato sifatida ko'rsatilmasligi kerak. Natijalar strukturali JSON shaklida olinishi mumkin: xato turi, koddagi joylashuvi, izoh, jiddiylik darajasi va taklif etilgan tuzatish.

Statik tahlil uchun CodeQL yoki C# ekotizimida ishlaydigan boshqa aniq vosita tanlanadi. Vosita nomi va versiyasi, yoqilgan qoidalar, SDK versiyasi hamda ishga tushirish buyrug'i hujjatlashtiriladi. Yakuniy natijalarni ko'rgandan keyin sozlamalarni o'zgartirish natijalarni xolis taqqoslashga xalaqit berishi mumkin.

3.4. Natijalarni birlashtirish qoidasi

Birlashtirishning sodda, tekshiriladigan varianti quyidagicha: (1) statik analizator va LLM topilmalarini yig'ish; (2) topilmalarni kod joylashuvi va xato turi bo'yicha moslashtirish; (3) bir xil xatoga tegishli takroriy topilmalarni birlashtirish; (4) faqat bitta usul bildirgan topilmalarni alohida belgilash; (5) etalon belgilar asosida barcha topilmalarni baholash. LLM va analizator bir xil xatoni turlicha ifodalashi mumkinligi sababli, moslashtirish faqat satrma-satr tenglik bilan emas, oldindan yozilgan qoida va inson tekshiruvi bilan amalga oshirilishi kerak.

Ushbu birlashtirish qoidasi dastlabki bazaviy variantdir. Uni yangi algoritm sifatida taqdim etishdan oldin mavjud adabiyotlarda o'xshash yondashuvlar tekshirilishi, keyin esa algoritmnining samaradorligi empirik natijalar bilan asoslanishi zarur.

3.5. Baholash mezonlari

Asosiy ko'rsatkichlar: $\text{precision} = \text{TP} / (\text{TP} + \text{FP})$, $\text{recall} = \text{TP} / (\text{TP} + \text{FN})$, $\text{F1} = 2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$. Bu yerda TP — to'g'ri topilgan etalon xato; FP — xato deb ko'rsatilgan, ammo tasdiqlanmagan holat; FN — mavjud bo'lgan, lekin aniqlanmagan xato. Ko'rsatkichlar bilan birga TP, FP va FN xom sonlari ham e'lon qilinishi kerak. Maxraj nol bo'lgan holatlarni hisoblash qoidasi oldindan belgilanadi.

Primary evaluation metrics (definitions, not measured results)

Precision	Recall	F1-score
$\text{TP} / (\text{TP} + \text{FP})$	$\text{TP} / (\text{TP} + \text{FN})$	$2\text{PR} / (\text{P} + \text{R})$
How many reported bugs are real?	How many known bugs were found?	Harmonic mean of precision and recall

2-rasm. Precision, recall va F1-score ko'rsatkichlarining ta'rifi.

Qo'shimcha ko'rsatkichlar sifatida har bir holatni tahlil qilish vaqti, foydalanish xarajati, jiddiylik darajasi bo'yicha moslik va taklif etilgan tuzatishning testlardan o'tish ulushi qayd etilishi mumkin. Natijalarni talqin qilishda tanlanma hajmi, ma'lumotlarning tarkibi va etalon belgilar sifati albatta hisobga olinadi.

3.6. Tushuntiruvchi misol

Quyidagi misolda nolga bo'lish ehtimoli ko'rsatilgan. Biroq bu kod faqat metod shartnomasi $\text{count}=0$ qiymatini taqiqlasa yoki bunday holat uchun xatti-harakat belgilanmagan bo'lsa, nuqson hisoblanadi. Tuzatishning to'g'ri varianti ilovaning talablariga bog'liq.

Example input (buggy C# code) <pre> public static int Average(int sum, int count) { return sum / count; // fails when count == 0 } </pre>	Expected safe revision (illustrative) <pre> public static int? Average(int sum, int count) { if (count == 0) return null; return sum / count; } </pre>
--	---

Illustrative example only; the appropriate fix depends on the application's contract.

3-rasm. C# kodidagi xatoni tushuntiruvchi sxematik misol; haqiqiy tajriba natijasi emas.

4. Kutilayotgan tahlil va natijalarni taqdim etish

Tajriba hali bajarilmagani sababli ushbu maqolada o'lchangan samaradorlik natijalari berilmaydi. Tajriba o'tkazilgach, quyidagi jadval haqiqiy qiymatlar bilan to'ldiriladi. Natijalar to'qib yozilmasligi va faqat yakuniy test to'plamidan olingan kuzatuvlarga asoslanishi shart.

Yondashuv	TP	FP	FN	Precision	Recall / F1
Faqat LLM	—	—	—	—	—
Faqat statik tahlil	—	—	—	—	—
LLM + statik tahlil	—	—	—	—	—

1-jadval. Tajriba natijalarini kiritish uchun shablon. Chiziqlar natija o'lchanmaganini bildiradi.

Natijalar bo'limida ma'lumotlar to'plamining hajmi va tarkibi, har bir usul topgan xatolar soni, precision/recall/F1, bajarilish vaqti va xarajatlar ko'rsatilishi kerak. Xatolar tahlilida kamida bir nechta noto'g'ri ijobiy va noto'g'ri salbiy misollar ko'rib chiqilishi tavsiya etiladi. Agar statistik test qo'llansa, uning tanlanish sababi va taxminlari izohlanadi.

5. Muhokama

Taklif etilayotgan metodologiyaning asosiy qiymati — LLM va statik tahlilni bir xil etalon to‘plamida baholash va ularning topilmalarini izchil moslashtirish imkoniyatidir. LLM kengroq kontekstni sharhlashi mumkin, statik analizator esa aniq qoidalarga tayangan holda barqaror ogohlantirishlar beradi. Biroq ushbu farqlar birlashtirilgan usul albatta ustun bo‘lishini anglatmaydi; bu faqat tajriba bilan tekshiriladigan taxmindir.

Natijalarga ta’sir qiluvchi asosiy cheklovlar quyidagilardan iborat: ma’lumotlar to‘plami barcha real dasturlarni ifodalamasligi; xato belgilarining to‘liq bo‘lmasligi; model versiyasi yoki provayder sozlamalarining o‘zgarishi; ochiq kodlar modelni o‘qitishda ishlatilgan bo‘lishi; hamda topilmalarni inson tomonidan moslashtirishdagi subyektivlik. Bu omillar maqolada ochiq ko‘rsatilishi va xulosalar tanlangan to‘plam doirasidan tashqariga haddan tashqari umumlashtirilmasligi kerak.

6. Xulosa

Maqolada C# dasturlaridagi xatolarni aniqlash uchun katta til modellari va statik tahlilni birlashtirishga mo‘ljallangan metodologik yondashuv bayon qilindi. Uchta konfiguratsiyani — faqat LLM, faqat statik tahlil va birlashtirilgan yondashuvni — bir xil etalon ma’lumotlar to‘plamida taqqoslash taklif etildi. Precision, recall, F1-score, bajarilish vaqti hamda xarajatlarni o‘lchash orqali usullarning afzallik va cheklovlarini baholash mumkin. Maqola metodologik taklif bo‘lgani uchun empirik ustunlik haqida xulosa bermaydi. Keyingi bosqichda ochiq C# ma’lumotlari asosida tajribani bajarish, natijalarni e’lon qilish va taklif etilgan birlashtirish qoidasining yangiligini mavjud tadqiqotlar bilan solishtirib asoslash zarur.

FOYDALANILGAN ADABIYOTLAR

- [1] GitHub. About CodeQL queries. CodeQL Documentation. <https://codeql.github.com/docs/writing-codeql-queries/about-codeql-queries/> (murojaat sanasi: 09.10.2026).
- [2] Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. R. (2024). SWE-bench: Can Language Models Resolve Real-world GitHub Issues? International Conference on Learning Representations (ICLR). <https://openreview.net/forum?id=VTF8yNQM66>
- [3] Microsoft. InferredBugs: A dataset of bugs inferred by static analysis. GitHub repository. <https://github.com/microsoft/InferredBugs> (murojaat sanasi: 09.10.2026).
- [4] CodeQL. C# query help. <https://codeql.github.com/codeql-query-help/csharp/> (murojaat sanasi: 09.10.2026).