VIEŠOJO NAUDOJIMO VERSIJA | PUBLIC USE VERSION

Contact & Attribution | Kontaktai

- **Author:** Arija A.
- **Email:** ari@ari.lt
- **Website:** https://ari.lt/

Note: This version has been anonymised to protect the privacy of the author, supervisors, and the educational institution. The findings, data analysis, and software tool ("CQaS") remain original intellectual property of the author.

Pastaba: Ši versija anonimizuota siekiant apsaugoti autorės, vadovų bei švietimo įstaigos privatumą. Išvados, duomenų analizė bei sukurtas programinis įrankis („CQaS") išlieka autorės nuosavybe.

[Perjungti į 1 skaidrę] Laba diena, gerbiama vertinimo komisija. Šiandien jums pristatysiu savo informatikos brandos darbą tema „Saugumo ir kodo kokybės balansavimas programavimo praktikose".

[Perjungti į 2 skaidrę] Šiuolaikiniame programavimo pasaulyje programuotojai nuolat susiduria su

dilema - kaip išlaikyti tinkamą balansą tarp programinio kodo saugumo ir jo kokybės. **[Pasirodo 1 punktas]** Mano darbo tikslas buvo nustatyti ir įgyvendinti programavimo praktiką (trumpai tariant, įvairių gairių ir rekomendacijų rinkinį), kuri veiksmingai suderina šiuos abu aspektus, taip užtikrinant ne tik patvaraus, bet ir patikimo bei ilgaamžio programinio kodo kūrimą. **[Pasirodo 2 punktas]** Iškelta problema yra itin aktuali: dažnai papildomos saugumo priemonės apsunkina kodo struktūrą, pablogina jo įskaitomumą ir gerokai apsunkina tolimesnę priežiūrą. **Be to,** šiandien kodo kūrimo procese vis dažniau naudojamas dirbtinis intelektas, pavyzdžiui vadinamasis „Vibe Coding", kas kelia papildomų nežinomųjų, susijusių su jo įtaka sistemos vientisumui.

[Perjungti į 3 skaidrę] Išanalizavus mokslinę literatūrą, išskyriau esminius principus, kurie padeda suvaldyti šią problemą. **[Pasirodo 1 punktas]** Pirma, simbiozė: būtina derinti kalibruotą automatizuotą analizę su atidžia ir koncentruota žmogaus peržiūra. **[Pasirodo 2 punktas]** Antra, matavimas: vertinant kelis aspektus sudėtingoje sistemoje, vieno rodiklio niekada nepakanka, todėl būtina stebėti kelis skirtingus matmenis. **[Pasirodo 3 punktas]** Trečia, sisteminimas: programavimo tyrimus ir darbus privalu atlikti sistemiškai, greitai, bei pasitelkiant aiškias ir pakartojimas gaires. **[Pasirodo 4 punktas]** Ir ketvirta, struktūra: norint sumažinti kognityvinį nuovargį skaitant ir prižiūrint kodą, privaloma skatinti išskaidymą į modulius.

[Perjungti į 4 skaidrę] Šie principai remiasi DevSecOps filosofiją - moderniu programinio kodo gyvavimo ciklo požiūriu. Kaip matote skaidrėje, šioje metodikoje saugumas (*Sec*) nėra paliekamas tik sistemos testavimo ar išleidimo etapui. Jis tampa integrali ir neatsiejama kiekvieno žingsnio dalimi - pradedant planavimu, baigiant kodo rašymu, testavimu bei sistemos stebėjimu.

[Perjungti į 5 skaidrę] Tam, kad galėčiau atlikti tyrimą ir objektyviai matuoti sistemų rodiklius, pritaikiau teorines žinias ir sukūriau atviro kodo įrankį, kurį pavadinau angliškų akronimu *Code Quality and Security* („CQaS"). **[Pasirodo 1 punktas]** Įrankis buvo sukurtas naudojant Python programavimo kalbą apimant virš 7000 kodo eilučių. Sukurtas įrankis apjungia kelis skirtingus analizės rodiklius. **[Pasirodo 2 punktas]** Jo pagrindinė funkcija - padėti prižiūrėti kodo sudėtingumą, jo įskaitomumą, oficialaus PEP8 stiliaus standarto atitikimą, apskaičiuoti apytikslę techninę skolą (*t.y. rodiklį, kuris įvardija sistemos struktūrinį sudėtingumą ir būsimų modifikacijų „sunkumo“ laipsnį*) ir net kodo dubliavimąsi. **[Pasirodo 3 punktas]** Šis įrankis palaiko tiek lietuvių, tiek anglų kalbas, todėl yra patogiai pritaikomas. Jis buvo sukurtas taip, kad jį galėtume ir ateityje praplėsti kitomis kalbomis, taikantis į globaliają rinką. **[Pasirodo 4 punktas]** Antrojoje versijoje (po eksperimentų) įdiegiau kodo peržiūros režimą pastebint teigiamą žmogaus peržiūros poveikį. Šis režimas ne tik atlieka statinę analizę, bet ir interaktyviai paprašo programuotojo atkreipti dėmesį į potencialiai įtartinas kodo vietas.

[Perjungti į 6 skaidrę] Kalbant apie patį eksperimentą, taikiau iteracinės daugiavariatinės analizės metodą. Užduotis buvo sukurti mikrotinklaraščių sistemą, ir iš viso eksperimentui sukūriau 13 skirtingų šios sistemos versijų apimančias beveik 13000 kodo eilučių. **[Pasirodo 1 eilutė: Bazinis kodas]** Pirmoji versija (1.0) buvo bazinis, nesaugus ir nekokybiškas kodas, kurį sukūrė žmogus (aš). Tai buvo atskaitos taškas. **[Pasirodo 2 eilutė: Tobulinimas]** Vėliau tyrėme kodo išskaidymą ir savarankišką tobulinimą, lygindami žmogaus darbą su dirbtiniu intelektu. **[Pasirodo 3 eilutė: Kontroliniai sąrašai]** Trečioje iteracijoje pritaikėme kontrolinius sąrašus - tiek pilnus, tiek

susiaurintus - vėlgi analizuodami žmogaus ir DI skirtumus. **[Pasirodo 4 eilutė: Automatinė analizė]** Ketvirtojoje dalyje žmogus taikė statinio ir dinaminio kodo analizės įrankius lyginant jų privalumus ir skirtumus. **[Pasirodo 5 eilutė: Kodo peržiūra]** Penktojoje iteracijoje vykdėme išsamią kodo peržiūrą ir taisymą, kur dar kartą lyginome žmogaus ekspertizę su DI. **[Pasirodo 6 eilutė: Dokumentacija]** Ir galiausiai įvertinome, ar dirbtinio intelekto sugeneruoti komentarai bei dokumentacija prideda apčiuopiamos naudos.

[Perjungti į 7 skaidrę] Rezultatai apie dirbtinio intelekto (konkrečiai „ChatGPT-5.2-Thinking" modelio) naudojimą buvo gana kategoriški. Nors šis modelis pasižymi aukštais samprotavimo gebėjimais, mūsų kontekste jo poveikis dažniausiai buvo neigiamas. **[Pasirodo 1 punktas]** Pagrindinė problema - DI nesugeba įvertinti pilno ir gilaus projekto konteksto. **[Pasirodo 2 punktas]** Pavedus DI atlikti kodo išskaidymą, sistema tapo fragmentuota, atsirado nesusipratimų tarp modulių. **[Pasirodo 3 punktas]** Taip pat, nors DI galbūt gali padėti parašyti greitą funkciją, integruojant tai į bendrą sistemą, procesas reikalavo itin kruopščios ir daug laiko atimančios žmogaus peržiūros. **[Pasirodo 4 punktas]** Priėjau išvados, kad DI įrankiai kol kas toli gražu negali pakeisti sistemingo žmogaus atliekamo darbo.

[Perjungti į 8 skaidrę] Surinkus duomenis iš visų versijų, atlikau kiekybinį rezultatų vertinimą. **[Pasirodo 1 punktas]** Norint palyginti labai skirtingas metrikas, pirmiausia rezultatus sunormalizavau naudojant min-max metodą. **[Pasirodo 2 punktas]** Toliau reitingavimui išnagrinėjau kelis statistinių vidurkių tipus ir pasirinkau triskaitį vidurkį (angl. Trimean). **Noriu pabrėžti vieną įdomų faktą apie šį terminą: ieškodama, kaip šią sąvoką taisyklingai pavadinti lietuviškai, radau tik 1 vienintelį šaltinį visoje literatūroje. Todėl, siekiant akademinio tikslumo, konsultavausi su Valstybine lietuvių kalbos komisija bei Vikipedijos bendruomene, kad oficialiai įtvirtinčiau „triskaičio vidurkio" sąvoką. Taip šiuo darbo metu gimė mano inicijuotas triskaičio vidurkio Vikipedijos straipsnis.** Šis vidurkio tipas buvo pasirinktas, nes jis idealiai subalansuoja skirtingus kodo kokybės ir saugumo rodiklius ir yra nepaprastai atsparus ekstremalioms reikšmėms imtyje nepamesdamas platesnio imties konteksto. **[Pasirodo 3 punktas]** Galiausiai tokiu būdu visi normalizuoti rezultatai buvo surikiuoti pagal optimalumą.

[Perjungti į 9 skaidrę] Šioje matricoje - eilutės atitinka įvairius matuotus rodiklius, o stulpeliai - versijas kurių metu buvo tiriamos tam tikros praktikos. Raudoni skaičiai viršuje rodo versijos reitingą, o antraštės dešinėje - įvarius matuotus rodiklius. Kuo langelio spalva tamsesnė, tuo rodiklis prastesnis; kuo šviesesnė - tuo geresnis.

Matricoje Galite pastebėti aiškius laimėtojus ir pralaimėtojus. Efektyviausios praktikos buvo **žmogaus vadovaujama kodo peržiūra (5.1 versija), žmogaus atliktas kodo išskaidymas (2.1 versija) ir statinis saugumo testavimas (4.1 versija)**. Prasčiausiai pasirodė **bazinė pirma versija ir beveik visos versijos, kurioms vadovavo dirbtinis intelektas**. Paskutinis stulpelis buvo **ne eksperimento dalis - tai, galutinis rezulttas palyginimui**, įrodantis šio brandos darbo sėkmingumą.

[Perjungti į 10 skaidrę] Remdamasi šia išsamia rodiklių analize, suformalavau efektyvią hibridinę

programavimo praktiką. Ją integravusi į savo įrankį ir sukūriau galutinę, 7-ąją sistemos iteraciją. **[Pasirodo 0 punktas]** Taikant naująją praktika ir sukurtą įrankį, rezultatai parodė įspūdingus rezultatus - 1-ąją vietą rezultatų matricoje. **[Pasirodo 1-2 punktas]** Palyginti su bazine versija, saugumo klaidų skaičius sumažėjo net 73 procentais. **[Pasirodo 3 punktas]** Apskaičiuota techninė kodo skola sumažėjo milžinišku skirtumu - daugiau nei 92 procentais. **[Pasirodo 4 punktas]** Be to, sistemoje buvo pasiektas visiškas, 100 procentų duomenų tipų užtikrinimas be jokių konfliktų - tai reiškia jog, pavyzdžiui, skaičius negalėtų būti sumaišytas su masyvu, kas yra rizika Python kalboje dėl jos duomenų tipų dinamiškumo. **[Pasirodo 5 punktas]** Bendrai, visų rodiklių vidutinis patobulėjimas siekė beveik 80%.

[Perjungti į 11 skaidrę] Apibendrinant šį darbą, atviro kodo bendruomenei ir sistemų kūrėjams teikiu šias mokslu ir praktika pagrįstas rekomendacijas, konstaduodama sudarytą praktiką: **[Pasirodo 1 punktas]** Išskaidymas: visuomet skaidykite kodą į nepriklausomus modulius. Tai ilgaamžiškumo pamatas. **[Pasirodo 2 punktas]** Sąrašai: aktyviai naudokite kontrolinius sąrašus, nes jie verčia programuotoją mąstyti sistemiškai ir nepraleisti kritinių taškų. **[Pasirodo 3 punktas]** Peržiūra: sisteminga, žmogaus vadovaujama kodo peržiūra vis dar yra viena efektyviausių praktikų kokybei užtikrinti. **[Pasirodo 4 punktas]** DI vengimas: kol kas siūlau vengti aklai pasikliauti dirbtinio intelekto priemonių išvestimi, ypač kai tai liečia sistemos architektūrą ir kriptografiją. **[Pasirodo 5 punktas]** Statinis Programų Saugumo Testavimas (arba SPST): naudokite automatizuotas statinės kodo analizės priemones, tokias kaip mano sukurtas „CQaS“ arba plačiau žinomus, tokius kaip „Bandit", kad automatiškai sulaikytumėte paviršutines klaidas anksčiau, nei jos pasiekia produkciją.

[Perjungti į 12 skaidrę] Darbo pabaigoje noriu tvirtai konstatuoti, jog iškeltas tikslas buvo pasiektas. **[Pasirodo 1 punktas]** Eksperimentiniu būdu įrodžiau ir nustačiau veiksmingą praktiką, kuri realiomis sąlygomis suderina kodo saugumą ir jo kokybę į vieną vienetą. **[Pasirodo 2 punktas]** Be to, šios praktikos ilgalaikiam ir automatizuotam palaikymui sėkmingai sukurtas atviro kodo įrankis.

[Perjungti į 13 skaidrę] Galiausiai, noriu išreikšti savo didžiulę padėką. **[Pasirodo 1 punktas]** Dėkoju savo brandos darbo vadovui (-ei), ... **[Pasirodo 2 punktas]** Taip pat esu labai dėkinga brandos darbo konsultantui (-ei) ...

[Perjungti į 14 skaidrę] Ačiū jums už dėmesį ir jūsų skirtą laiką. Dabar mielai atsakysiu į visus jūsų klausimus.