

UPOTREBA ALATA ZA STATIČKU ANALIZU KODA U UČENJU

Danilo Nikolić¹, Miroslav Stefanović², Andrea Gutai³, Ivana Spasojević⁴, Teodora Lolić⁵

^{1,2,3,4,5} Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, Srbija

¹nikolic.danilo@uns.ac.rs

Kratak sadržaj: Alati za statičku analizu koda sve se više koriste za poboljšanje kvaliteta koda. Ovi alati vrše analizu izvornog koda programa i detektuju greške, ranjivosti, sigurnosne nepravilnosti i slično. Kvalitet izvornog koda je jedan od ključnih faktora uspešnosti i kvaliteta svakog softverskog proizvoda i zahteva konstantan nadzor.. Alati za statičku analizu koda omogućavaju programerima da na lakši način uoče moguće nepravilnosti u kodu i otklone ih. Cilj ovog rada je da istakne mogućnost primene statičke analize koda u obrazovanju studenata u oblasti informacionih tehnologija. Koristeći sve mogućnosti koje pruža statička analiza, moguće je kreirati digitalne asistente uključene u razvojno okruženje koji treba da pomognu studentima da uče da pišu kvalitetniji kod.

Ključne reči: Statička analiza koda, Alati, Digitalni asistenti

USE OF STATIC CODE ANALYSIS TOOLS IN LEARNING

Abstract: Static code analysis tools are increasingly being used to improve code quality. These tools analyze the source code of the program and detect errors, vulnerabilities, security flaws etc. The quality of the source code is a key factor of any software product and requires constant monitoring. Static code analysis tools allow programs to more easily spot and eliminate possible code irregularities. The aim of this paper is to emphasize the possibility of applying static code analysis in student education in the field of information technology. Using all the possibilities provided by static analysis, it is possible to create digital assistants involved in the development environment, which should help students learn to write better quality code.

Key Words: Static code analysis, Tools, Digital assistants

1. UVOD

Kvalitet izvornog koda je ključni faktor svakog softverskog proizvoda i zahteva stalnu verifikaciju i nadzor. Statička analiza se koristi za održavanje i poboljšanje kvaliteta izvornog koda. Statička analiza programskog koda koristi se od ranih 1960-ih za optimizaciju rada kompajler [1]. Kasnije se pokazala korisnim za alate za otklanjanje grešaka, kao i za razvojna okruženja. Sve je veći broj alata koji omogućavaju statičku analizu koda, od kojih su mnogi alati otvorenog koda (eng. *Open source*) i omogućavaju analizu nekoliko različitih programskih jezika [2].

Razvoj softvera sa nula grešaka predstavlja nerealan cilj. Korišćenje alata za statičku analizu u procesu softverskog inženjerstva može pomoći u smanjenju broja softverskih nedostataka. Nažalost, alati često prijavljuju veliki broj suvišnih upozorenja poznatih kao lažno pozitivne greške (eng. *False Positives*) koji ometaju usvajanje alata. Primena ovih alata na postojećim velikim bazama kodova može još više povećati broj lažno pozitivnih rezultata. Većina alata pronalazi slične nepravilnosti, ali postoje i različiti skupovi i kategorije nepravilnosti koje različiti alati ističu, to sve zavisi od programskih jezika za koje obezbeđuju podršku i svrhe za koju su alati namenjeni.

Alati za statičku analizu postoje u mnogim oblicima koji ispunjavaju različite svrhe. Neki alati su specijalizovani za pronalaženje bezbednosnih propusta, dok drugi analiziraju kod zbog stilskih problema. Mnogo različitih delova softvera može se pratiti i analizirati pomoću ovih alata. U modernim kompajlerima, tipovi, a ponekad i vrednosti se prate i proveravaju u odnosu na unapred određena ograničenja i zahteve. To omogućava kompajlerima da izvrše agresivnu optimizaciju prateći konstantne vrednosti vremena kompajliranja i nedostupan kod. Napredni alati za statičku analizu teže da naprave još temeljnije analize od kompajlera. Kako neki oblici analize mogu potrajati prilično dugo, u zavisnosti od veličine baze koda, korisno je odvojiti je od kompajliranja.

Ostatak rada organizovan je na sledeći način. Odeljak 2 predstavlja postojeća povezana istraživanja u ovoj oblasti. Odeljak 3 opisuje se statička analiza koda i alati koji su razvijeni. U odeljku 4 predstavljen je koncept digitalnih asistenata upotrebom statičke analize koda, odeljak 5, predstavlja zaključak istraživanja. Na kraju, odeljak 6 predstavlja reference korišćene u radu.

2. STANJE U OBLASTI

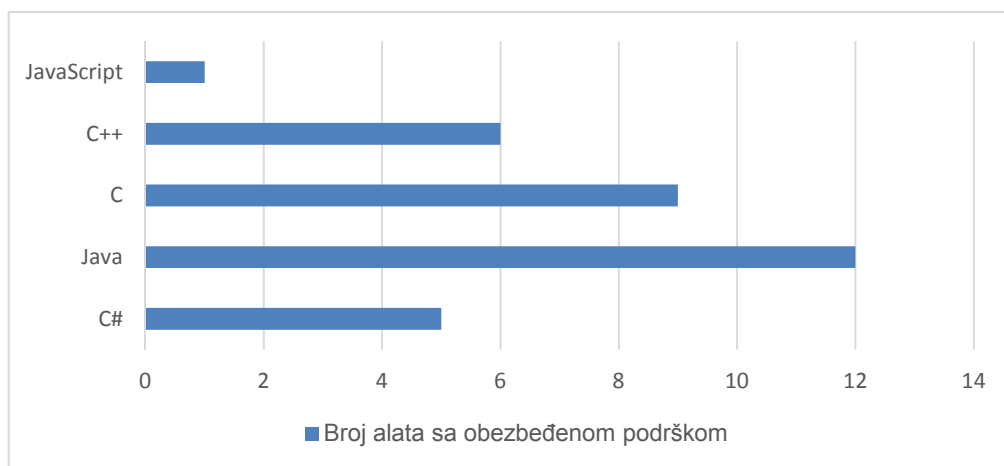
Kada je reč o alatima za statičku analizu koda, veliki problem predstavlja činjenica da postoji veliki broj alata koji pružaju ovu vrstu analize. Broj alata se neprestano povećava i potrebno je klasifikovati postojeće alate prema programskom jeziku za koji imaju podršku i vrstama nedostataka koje otkrivaju.

Analiza statičkog koda vrši se na izvornom kodu programa i zato su programski jezici na kojima je program napisan veoma važni.

Spoveden je sistematski pregled literature stanja u oblasti. Rad koji opisuje proces sistematskog pregleda literature za potrebe ovog rada opisan je u okviru [3]. Studije koje su obuhvaćene pregledom literature postavljaju se na određeno vreme, u ovom pregledu obuhvaćeni su studije objavljene u prethodnih deset godina. Pregled stanja u oblasti je obavljen na osnovu smernica koje je za sistematski pregled literature iz softverskog inženjerstva izvela Barbara Kitchenham [4], sa fokusom na prikazivanje najčešće korišćenih alata podeljenih programskim jezicima za koje imaju obezbeđenu podršku i nedostatke koje detektuju.

Odnosno pregledom dobijenih naslova, zatim apstrakta i samog sadržaja rada u pregled uključeno je 22 primarne studije, neke od reprezentativnih primera uključenih studija su [5, 6, 7, 8]. Detaljna procedura ovog sistematskog pregleda literature po koracima predstavljena je u okviru [3].

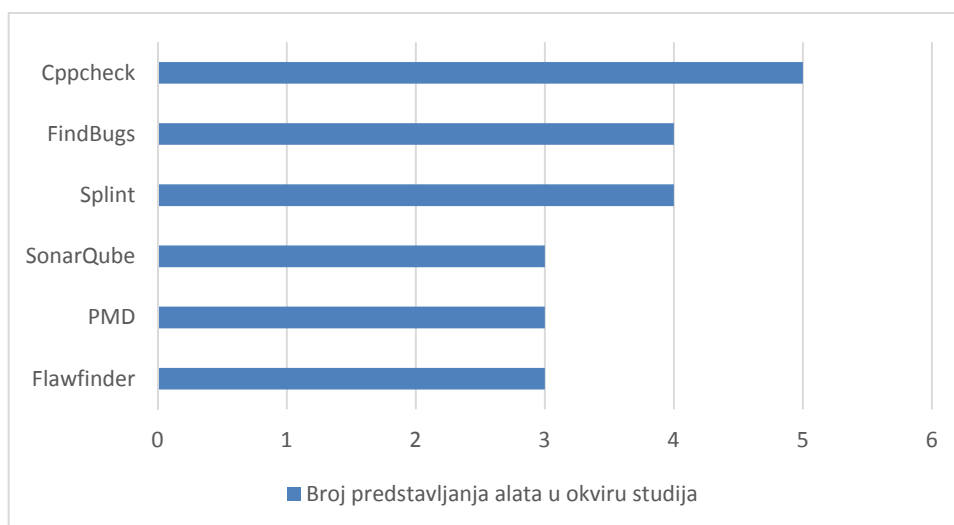
Najčešće se u oblasti statičke analize vrši istraživanje korišćenjem alata nad programskim jezicima opšte namene, odnos između alata sa obezbeđenom podrškom za određene programske jezike predstavljen je na slici 1.



Slika 1. Broj alata sa obezbeđenom podrškom za programske jezike [3]

Pored ove analize, posmatrano je i koji alati za statičku analizu koda se najčešće koriste u istraživanjima i koji alati se analiziraju na osnovu vrste nepravilnosti koje detektuju i programskih jezika za koje imaju obezbeđenu podršku.

Predstavljeno je šest alata koji su najčešće analizirani i predstavljeni u primarnim studijama. To su alati koji su tri ili više puta predstavljeni u različitim istraživanjima. U poređenju sa tim, drugi alati predstavljeni u okviru studija obuhvaćenih ovim pregledom analizirani su u dve ili jednoj studiji. Grafički predstavljeni rezultati prikazani su na slici 2.



Slika 2. Najčešće predstavljeni alati u okviru studija [3]

3. ALATI ZA STATIČKU ANALIZU KODA

Pregled koda se koristi za poboljšanje procesa održavanja softverskog proizvoda, povećavajući njegovu pouzdanost i sigurnost. Aktivnost pregleda koda može smanjiti broj potrebnih ispravki u izvornom kodu i može se lako integrisati u životni ciklus razvoja softvera, tako da iskusniji softver developeri često sprovode ovu aktivnost u svom radu. Međutim, ručni pregled izvornog koda traje veoma dugo [9].

Da bi izvršili efikasan ručni pregled koda, recenzenti moraju znati sve moguće nedostatke i netačnosti pre nego što pažljivo provere sav izvorni kod. Kao zamena za duži ručni pregled izvornog koda, preporučuje se upotreba alata za statičku analizu koda. Ovi alati su brži i sadrže znanje za obavljanje pregleda izvornog koda. Međutim, ovi alati ne mogu u potpunosti zameniti ručni pregled, ali olakšavaju ovaj proces.

Proces statičke analize koda koristan je ne samo za optimizaciju rada kompajlera (što je bila prvobitna namena) već i za otkrivanje nepravilnosti i mogućih nedostataka. Na ovaj način je moguće stvoriti alate koji će programerima pomoći da razumeju ponašanje programa i da identifikuju razne nedostatke programa bez njegovog izvršenja. Alati koji se koriste za statičku analizu koda su programi koji objašnjavaju ponašanje drugih programa [1].

Statička analiza koda može biti izvršena nad izvornim kodom programa i bez alata za statičku analizu, taj proces se naziva manualni pregled koda. Međutim, ovi alati su razvijeni kako bi se ova aktivnost pregleda automatizovala i učinila lakšom. Od kada se izučava statička analiza koda razvijen je veliki broj alata za statičku analizu koda, ovi alati se svakodnevno unapređuju, povećava se broj jezika za koje imaju obezbeđenu podršku, smanjuje se vreme potrebno za izvršenje statičke analize koda i smanjuje se broj grešaka alata (lažno pozitivni i lažno negativni rezultati analize).

Pored razvoja i unapređenja postojećih alata, razvijaju se i novi alati za statičku analizu i svakodnevno se broj dostupnih alata za statičku analizu uvećava.

4. DIGITALNI ASISTENTI

Statička analiza koda omogućava sprovođenje analize nad izvornim kodom programa i generisanje izveštaja o detektovanim nepravilnostima u programu. Moguće je izvršiti klasifikaciju procesa statičke analize koda u zavisnosti da li se ona sprovodi u toku razvoja softverskog rešenja ili neposredno nakon završetka razvoja.

Tako, moguće je razlikovati dve osnovne strategije sprovođenja statičke analize, to su [10]:

1. Pokretanje alata za statičku analizu nad završenim softverskim projektom.
2. Pokretanje alata za statičku analizu u toku razvoja softverskog projekta.

Sprovođenje ideje digitalnih asistenata u obrazovanju moguće je jedino u okviru druge strategije, odnosno ukoliko se statička analiza sprovodi u toku razvoja softverskog rešenja.

Na taj način, upotrebom veštačke inteligencije ovi alati uključeni u razvojno okruženje u kojem se razvija program, vršili bi automatsku analizu napisanog koda i generisali moguće nepravilnosti i nedostatke u izvornom kodu programa. Alati za statičku analizu koda, sprovode analizu pretvaranjem koda u običan tekst i upoređivanjem napisanog koda sa pravilima i standardima koji postoje za određeni programski jezik. Na taj način, na sličan način na koji to radi kompajler, korisnicima ovih alata prijavljivali bi uočene nepravilnosti, sigurnosne ranjivosti, sigurnosne tačke i druge vrste nepravilnosti specifične za određeni programski jezik.

Trenutno, postoji veliki broj alata za statičku analizu koda, neki od tih alata omogućavaju njihovo uključivanje u različita razvojna okruženja u cilju podrške prilikom razvoja softverskog rešenja. Ipak, razvoj digitalnih asistenata u razvojnim okruženjima korišćenjem statičke analize koda, podrazumevalo bi da u razvojnom okruženju postoji čet-bot (eng. *Chatbot*) [11] koji obavlja komunikaciju sa korisnikom na osnovu sprovedene analize nad napisanim izvornim kodom. Na taj način, digitalni asistenti bi direktno učestvovali u procesu razvoja softverskog rešenja korisnika i savetovali kako da struktuirati i napiše određeni deo koda, takođe, ovi asistenti ukazivali bi na određene nepravilnosti i defekte u kodu i nalazili načine za njihovo otklanjanje.

Kao još jednu od funkcionalnosti digitalnih asistenata treba uključiti i sisteme preporuke (eng. *Recommender system*), to je sistem koji je u stanju da predvidi buduću aktivnost korisnika i da mu iz skupa određenih mogućnosti rešenja predloži one koje odgovaraju njegovim preferencijama. Ovi sistemi treba da posluže i da navode korisnike prilikom donošenja odluke na koji način da ispravi određenu nepravilnost ili da razvije određeni deo koda, na taj način oni poboljšavaju zadovoljstvo korisnika prilikom upotrebe razvojnog okruženja.

Alati za statičku analizu koda koriste se kako u industriji tako i u obrazovanju. Posebno je važna uloga ovih alata u obrazovanju studenata u oblasti informacionih tehnologija, razlog tome je što ovi alati ukazivanjem na određene nepravilnosti i defekte u izvornom kodu, uče studente kako da pišu kvalitetniji kod i stvaraju bolja softverska rešenja. Razvoj digitalnih asistenata, omogućio bi još bolju podršku učenju i razvoju budućih inženjera informacionih tehnologija. Uključivanje digitalnih asistenata, u toku učenja kod studenata, treba da olakša proces učenja, obezbedi učenje pisanja kvalitetnog koda i generiše sisteme preporuke za rešavanje problema na koje studenti nailaze prilikom razvoja projekta u toku studija.

5. ZAKLJUČAK

Alati za statičku analizu koda sve više nalaze svoju primenu i sve više se koriste u svakodnevnom radu softver developera na različitim pozicijama. Takođe, ovi alati, još uvek ne u potpunosti, pronalaze svoju primenu i u edukaciji budućih softver developera o potencijalnim nepravilnostima koje mogu nastati u kodu, kao i o njihovom otklanjanju reinženjeringom izvornog koda.

Svakodnevno se razvijaju novi i unapređuju postojeći alati, nudeći svojim korisnicima sve kvalitetniju uslugu stvaranjem poverenja u rezultate statičke analize i omogućavajući im da na kvalitetan način izbegnu određene nepravilnosti u kodu i učine svoje softversko rešenje kvalitetnijim. Ovo je posebno važno, iz razloga što je još uvek manja upotreba alata za statičku analizu koda uzrokovana upravo nepoverenjem korisnika u ove alate i vreme koje je potrebno utrošiti za sprovođenje ovakvih analiza nad izvornim kodom programa. Kada se govori o nepoverenju u rezultate statičke analize, oni se prvenstveno odnose na generisanje lažno pozitivnih i lažno negativnih rezultata. Razvijanjem i unapređenjem ovih alata, procenat grešaka se umanjuje i ovi alati daju bolje rezultate, samim tim se i više koriste.

Razvoj digitalnih asistenata i njihovo uključivanje u nastavu u oblasti informacionih tehnologija predstavlja sledeći korak u razvoju kvaliteta nastave na fakultetima u ovoj oblasti. Ovi asistenti omogućili bi studentima sugestije za pisanje kvalitetnije koda i predlagali načine za otklanjanje određenih nepravilnosti u kodu.

Ipak, pre nego se uključe u proces nastave na fakultetima, neophodno je sprovesti različita istraživanja i eksperimente nad ovim asistentima i proveriti na koji način oni utiču na korisnike i da li daju očekivane rezultate kod njihovih korisnika.

ZAHVALNICA

Ovaj rad je podržan od strane Fakulteta tehničkih nauka u Novom Sadu, Departmana za industrijsko inženjerstvo i menadžment, u okviru realizacije projekta pod nazivom: "Istraživanje mogućnosti primene veštačke inteligencije u industrijskom inženjerstvu, menadžmentu i nastavnom procesu na DIIM.

6. LITERATURA

- [1] A. Moller and I. Schwartzbach, *Static program analysis*, 2019.
- [2] M. Beller, R. Bholanath and S. M., "Analyzing the state of static analysis: A large-scale evaluation in open source software," pp. 470-481, 2016.
- [3] D. Stefanović, D. Nikolić, D. Dakić, I. Spasojević and S. Ristić, "Static code analysis: a systematic literature review," in Proceedings of the 31st DAAAM International Symposium (u pripremi), Vienna, 2020..
- [4] B. Kitchenham, *Procedure for Undertaking Systematic Reviews*, Keele University: Computer Science Department, 2004.
- [5] A. Arusoai, S. Ciobaca and V. Cracion, "A comparison of open-source static analysis tools for vulnerability detection in C/C++ Code," in Proceedings - 2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, 2018.
- [6] M. Abdul Bari and S. Ahamad, "Code Cloning: The Analysis, Detection and Removal," International Journal of Computer Applications, vol. 20, no. 7, pp. 34-38, 2011.
- [7] A. Kaur and R. Nayyar, "A Comparative Study of Static Code Analysis tools for Vulnerability Detection in C/C++ and JAVA Source Code," Procedia Computer Science, vol. 171, no. 2019, pp. 2023-2029, 2020.
- [8] C. Chahar, V. Chauhan and M. Lalik, "Code Analysis for Software and System Security Using Open Source Tools," Information Security Journal, vol. 21, no. 6, pp. 346-352, 2012.
- [9] J. West and B. Chess, "Secure programming with static code analysis," Pearson education, 2007.
- [10] J. Xie, H. R. Lipford and B. Chu, "Evaluating interactive support for secure programming," Conference on Human Factors in Computing Systems, pp. 2707-2716, 2012..
- [11] D. Lalić, N. Krstić and S. Havzi, "THE BUSINESS IMPLICATIONS OF CHATBOTS," 2019.
- [12] A. Mamun, H. Khanam and H. Grahn, "Comparing Four Static Analysis Tools for Java Concurrency Bugs," in Third Swedish Workshop on Multi-core Computing (MCC '10), 2010.
- [13] K. Goseva-Popstojanova and A. Perhinschi, "On the capability of static code analysis to detect security vulnerabilities," Information and Software Technology, vol. 68, pp. 18-33, 2015.
- [14] P. Nunes, I. Medeiros and J. Fonseca, "An empirical study on combining diverse static analysis tools for web security vulnerabilities based on development scenarios," Computing, vol. 101, no. 2, pp. 161-185, 2019.