Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004:007
DOI: https://doi.org/10.24867/18OI01Stankovic

KOMPARATIVNA ANALIZA ALATA ZA STATIČKU ANALIZU KODA U SVRHU IDENTIFIKACIJE I PROCENE TEHNIČKOG DUGA U .NET PROJEKTIMA

COMPARATIVE ANALYSIS OF TOOLS FOR STATIC CODE ANALYSIS FOR THE PURPOSE OF IDENTIFICATION AND ESTIMATION OF TECHNICAL DEBT IN .NET PROJECTS

Milica Stanković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – INŽENJERSTVO INFORMACIONIH SISTEMA

Kratak sadržaj – *U radu je predstavljeno istraživanje kojim se ispituju postojeći alati za statičku analizu koda s ciljem identifikacije i estimacije količine tehničkog duga u .NET projektima. Predmet istraživanja su alati: SonarCloud, Squore, NDepend i Sonargraph. Cilj istraživanja je da se uporedi dobijena količina tehničkog duga u izvornom kodu kako bi se dobila slika preciznosti (pouzdanosti) estimacija tehničkog duga koje navedeni alati pružaju.*

Ključne reči: *TD (engl. Technical debt), Alati za statičku analizu koda*

Abstract – *This research examines the existing tools for static code analysis with the aim of identifying and estimating the amount of technical debt in .NET projects. The tools that were tested were: SonarCloud, Squore, NDepend and Sonargraph. The aim of the research is to compare the obtained amount of technical debt in the source code in order to get a picture of the precision (reliability) of the estimation of technical debt that these tools provide.*

Keywords: *Technical debt, Static code analysis tools*

1. UVOD

Tehnički dug (TD, engl. *Technical debt*) je termin koji se koristi za opisivanje odloženih troškova održavanja, uzrokovanih početnim kompromisom između kvaliteta i brzine [1]**.** Velike kompanije troše i do 41% svog budžeta za informacione tehnologije (IT) na TD, dok male troše oko 27%. Neke studije daju procene da se na probleme izazvane tehničkim dugom troši između 2 i 20 američkih dolara po liniji izvornog koda. TD je skriveni trošak, koji utiče na kvalitet softvera, profit kompanije, ali i na zadovoljstvo korisnika. Da bi se izbegla prevelika akumulacija tehničkog duga, pre svega potrebno je identifikovati dug i njegove uzroke, a zatim i adekvatno upravljati TD-om [2]. Na tržištu su dostupni različiti alati za statičku analizu koda (ASAT, engl. *ASAT tools*) koji olakšavaju proces razvoja i uočavaju različite defekte u kodu. Statička analiza se može koristiti za detekciju TD-a, pronalaženje grešaka, proizvodnju efikasnijeg koda, zatim u svrhe otklanjanja grešaka, analizu uticaja, otkrivanje upada (engl. *intrusion detection*) i za softverske metrike.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Mandić.

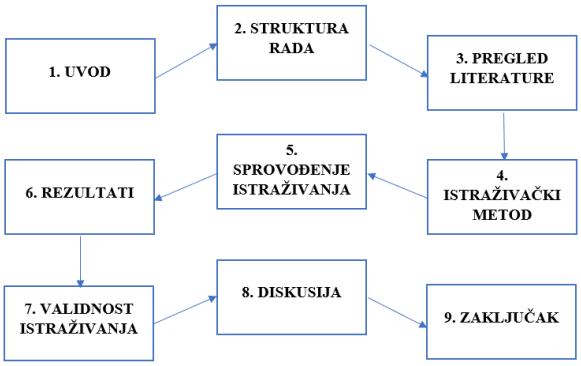
U ovom radu je glavni fokus na funkcionalnostima ASAT alata koje pružaju identifikaciju i estimaciju tehničkog duga.

Cilj ovog istraživanja jeste ispitivanje preciznosti, odnosno pouzdanosti estimacija tehničkog duga koje pružaju sledeći ASAT alati: *SonarCloud*[1], *NDepend*[2], *Squore*[3] i *Sonargraph*[4]. ASAT alati skeniraju izvorni ili binarni kod u odnosu na skup unapred definisanih problema. Mogu se konfigurisati da otkriju funkcionalne probleme (kao što su curenje resursa ili netačna logika) i probleme sa održavanjem (kao što su neusklađenost sa najboljim praksama ili kršenja stilskih konvencija). Koriste tehnike kao što su analiza protoka podataka i analiza toka kontrole, kako bi pronašli defekte u kodu [3]. Generišu metrike koje se odnose na održivost, pokrivenost koda testovima, duplikacije koda, veličinu aplikacije, kompleksnost, probleme i slično.

Takođe, generišu indekse koji se odnose na tehnički dug, kao što je, na primer *Structural Debt Index*. Ovi alati na objektivan način pomažu praćenje kvaliteta aplikacije, bilo u fazi razvoja ili održavanja.

2. STRUKTURA RADA

Rad se sastoji iz devet poglavlja, a svako poglavlje obrađuje određenu temu. Struktura rada prikazana je na Slici 1.



Slika 1. *Struktura rada*

[1] https://sonarcloud.io/
[2] https://www.ndepend.com/
[3] https://www.squoring.com/
[4] https://www.hello2morrow.com/products/sonargraph

3. PREGLED LITERATURE

U ovom poglavlju biće reči o *code smells, SQALE* modelu za rangiranje kvaliteta softvera, estimaciji TD-a i tipovima TD-a.

3.1 Identifikacija *Code smells*

Code smells su karakteristike softvera koje mogu da prouzrokuju problem u kodu ili dizajnu, i da otežaju dalji razvoj i održavanje softvera. Za identifikaciju *code smells* najbolje je koristiti ASAT alate, iz razloga što sami programeri mogu da propuste mnoge *code smells* u toku rada [4]. *ASAT* alati koriste baze znanja kako bi identifikovali probleme.

Baza znanja predstavlja skup definisanih pravila u odnosu na koje će se vršiti provera koda. U većini alata baza znanja može da se proširi ili izmeni, u skladu sa potrebama korisnika. Ukoliko se statičkom analizom utvrdi da je narušeno određeno pravilo iz baze znanja, većina alata će ispisati koje pravilo je narušeno, u kom delu koda, i daće uputstvo kako da se greška ispravi.

3.2 *SQALE* model za rangiranje kvaliteta softvera

Kvalitet softvera može da se definiše kao usklađenost sa eksplicitno navedenim funkcionalnim zahtevima i zahtevima performansi, eksplicitno dokumentovanim razvojnim standardima i implicitnim karakteristikama koje se očekuju od svih profesionalno razvijenih softvera [5].

SQALE model predstavlja generički pristup za modeliranje kvaliteta softvera i može se primeniti na bilo koji programski jezik. Implementiran je od strane mnogih ASAT alata, među kojima su i *SonarQube* i *Squore*. Pristup je zasnovan na 8 karakteristika koda koje su hronološki organizovane u formi piramide. Na dnu piramide je mogućnost testiranja (engl. *testability*), a zatim iznad slede pouzdanost, promenljivost, efikasnost, bezbednost, mogućnost održavanja, prenosivost, i na vrhu piramide, mogućnost ponovne upotrebe (engl. *reusability*). Zahtevi kvaliteta kao što je "*Exception Handling shall not catch Null Pointer Exception*" su povezani sa karakteristikama u piramidi i imaju cenu sanacije. Ako zahtev za kvalitetom utiče na više od jedne karakteristike, formira se asocijacija sa najnižom karakteristikom. Karakteristike na dnu piramide predstavljaju važnije dimenzije kvaliteta i imaju za cilj da pomognu programerima kada daju prioritet zahtevima koje je potrebno refaktorisati [6].

3.3 Estimacija tehničkog duga

U ovoj sekciji biće predstavljen model za estimaciju TD-a koji je implementiran u *SonarQube TD-Plugin*. Ovaj model koristi sledeće formule kako bi izračunao vrednost tehničkog duga:

1. *Debt = duplication + violations + comments + coverage + complexity + design*
2. *duplication = cost_to_fix_one_block * duplicated_blocks*
3. *violations = cost_to_fix_one_violation * mandatory_violations*
4. *comments = cost_to_comment_one_API * public_undocumented_API*
5. *coverage = cost_to_cover_one_of_complexity * uncovered_complexity_by_tests*
6. *design = cost_to_cut_an_edge_between_two_files * package_edges_weight*
7. *complexity = cost_to_split_a_method * (function_complexity_distribution ≥ 8) + cost_to_split_a_class * (class_complexity_distribution ≥ 60)* [7].

Prethodno navedene metrike *duplication, violations, comments, coverage, complexity* i *cycles* se izražavaju u čovek-danima (1 dan = 8h). U nastavku će detaljnije biti objašnjena svaka od prethodno navedenih metrika Metrika *duplication* se odnosti na procenjeni napor koji je potreban za ukljanjanje dupliciranog koda. *Violations* je procenjeni napor koji je povezan sa uklanjanjem narušavanjima, prekršajima (engl. *violations*) u sistemu. *Coverages* predstavlja procenjeni napor potreban da se kod pokrije 80% sa testovima. *Complexity* je ukupan procenjeni napor potreban da se svaka metora i klasa podeli (engl. *split*). *Comments* predstavlja procenjeni napor povezan sa dokumentovanjem API-ja (engl. *Application Programming Interface*). *Design* se odnosi na procenjeni napor povezan sa sečenjem svih postojećih ivica između datoteka (engl. *cutting all existing edges between files*). Svaka od navedenih metrika je izražena u čovek-danima. Kako bi se procenilo trošak tehničkog duga po čovek-danu, primenjuje se finalna kalkulacija upotrebom podrazumevane vrednosti od 500$ [7].

3.4 Tipovi tehničkog duga

U ovoj sekciji biće navedeni tipovi tehničkog duga koji su identifikovani na osnovu detaljnog istraživanja u radu "*Towards an Ontology of Terms on Technical Debt*". To su sledeći tipovi TD-a: *Architecture Debt, Build Debt, Code Debt, Defect Debt, Documentation Debt, Design Debt, Infrastructure Debt, People Debt, Process Debt, Requirement Debt, Service Debt, Test Automation Debt, Test Debt* [8].

4. ISTRAŽIVAČKI METOD

Za formulisanje cilja istraživanja i istraživačkih pitanja korišćena je poznata metoda GQM (engl. *Goal/Question/-Metric*) [9]. Ciljevi će biti predstavljeni u Tabeli 1.

Tabela 1 – *Istraživački Ciljevi*

	Cilj 1	**Cilj 2**
Predmet	Alat za statičku analizu koda	Alat za statičku analizu koda
Svrha	Identifikacija (karakterizacija)	Estimacija
Fokus	Tehnički dug (TD)	Količina TD-a u izvornom kodu
Stanovište	Istraživač	Istraživač

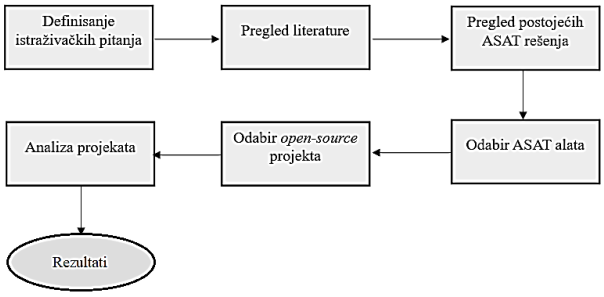
GQM metoda nalaže da se za svaki cilj postavi jedno ili više pitanja koja će pomoću da se okarakteriše procena ili postizanje navedenog cilja. U skladu sa tim definisana su sledeća dva pitanja na koja je potrebno odgovoriti, i koja u isto vreme predstavljaju i istraživačka pitanja vezana za prvi cilj.

1. *Koje vrste tehničkog duga alat može da prepozna?*
2. *Koje modele identifikacije alat koristi?*

Pitanja vezana za drugi cilj su:

1. *Koje metrike i modeli za estimaciju tehničkog duga se koriste?*
2. *Koja je preciznost (pouzdanost) estimacija tehničkog duga?*

Na Slici 2 prikazani su metodološki koraci istraživanja.



Slika 2. *Metodološki koraci istraživanja*

5. SPROVOĐENJE ISTRAŽIVANJA

Od ukupno 12 alata koji su pronađeni pregledom literature i pregledom postojećih *ASAT* rešenja, odabrano je 4 alata koji su zadovoljili postavljene kriterijume (podržavaju .NET tehnologiju i imaju besplatnu licencu određeno vreme). Odabrani su alati *SonarCloud*, *NDepend*, *Squore* i *Sonargraph.* Od potencijalnog skupa projekata sa *GitHub* platforme odabrano je 6 projekata koji su ispunili kriterijume (razvijeni u *.NET* tehnologiji, imaju minimum 20.000 linija koda). Odabrani projekti su: *OrchardCore, nopCommerce, eShopOnContainers, clean-architecture-manga, Virto Commerce i SimplCommerce.*

Da bi se izvršilo testiranje projekata u odabranim alatima, najpre je bilo potrebno izvršiti odgovarajuća podešavanja u svakom od alata. *SonarCloud* je besplatan za sve *open source* projekte. Veoma lako se može povezati sa *GitHub* nalogom, na kojem su *fork*-ovani projekti za testiranje. Bilo je potrebno uvesti organizaciju iz *GitHub*-a u *SonarCloud*, što je i urađeno jer se potrebni projekti svakako nalaze na *GitHub*-u. Sledeće što je bilo potrebno je kreiranje projekta koji će biti analiziran u *SonarCloud*-u. Dalja podešavanja u *SonarCloud* alatu nisu bila potrebna iz razloga što je za testiranje korišćen *Azure DevOps* servis. Što se tiče postavki na *Azure DevOps* platformi, pre svega je bilo potrebno ulogovati se na platformu korišćenjem *Microsoft* naloga.

Nakon toga, bilo je potrebno kreirati organizaciju na *Azure DevOps* nalogu. Kada je kreirana organizacija, može se pristupiti kreiranju projekata. Da bi se izvršilo testiranje, potrebno je za svaki projekat kreirati i podesiti *pipeline*. Zatim je bilo potrebno povezati *Azure DevOps* sa *GitHub*-om, *SonarCloud*-om i dodati *NDepend* ekstenziju. Probna licenca za *NDepend* važi 28 dana. Besplatna licenca za *Squore* i *Sonargraph* traje 14 dana. Pre početka rada sa ovim alatima trebalo je nabaviti licence. Postavke u *Squore* i *Sonargraph* alatu su relativno jednostavne, odnosno u samo par klikova testiranje može da se pokrene. Za analizu u *Squore* i *Sonargraph* alatu sa *GitHub*-a su preuzeti projekti u .zip formatu a u alatima je dodata putanja do preuzetih projekata. Tokom rada sa *Squore* i *Sonargraph* alatom, uočeno je da ovi alati imaju veoma dobru korisničku podršku.

6. REZULTATI

Prvo pitanje vezano za cilj 1 glasilo je: *koje vrste tehničkog duga alat može da prepozna?* Odgovor na ovo pitanje sledi u Tabeli 2.

Tabela 2. *ASAT Alati i vrste T.D.-a*

***ASAT* alat**	**Vrsta tehničkog duga**
SonarCloud	*Code debt*
NDepend	*Architectural debt, design debt, code debt*
Squore	*Design debt, code debt*
Sonargraph	*Architectural debt, design debt*

Drugo pitanje vezano za Cilj 1 glasilo je: *koje modele identifikacije alat koristi? Odgovor na ovo pitanje sledi u Tabeli 3.*

Tabela 3. *Modeli Identifikacije*

ASAT alat	***Model identifikacije***
SonarCloud	*SQALE metodologija*
NDepend	*SQALE metodologija*
Squore	*SQALE metodologija*
Sonargraph	*Primenjuje skup od 117 metrika (metrike nezavisne od programskog jezika + C# metrike) koje su definisane od strane Sonargraph-a.*

Prvo pitanje vezano za Cilj 2 glasi: *koje metrike i modeli za estimaciju TD se koriste? SonarCloud* koristi metrike *Debt, Debt ratio, Rating, Effort to reach A. NDepend* koristi sledeće metrike: *Debt, Issues, Annual Interest, Breaking Point, Lines of code, DebtRating, DebtRatio. Squore koristi metrike: Technical Debt, Source Lines of Code, Key performance indicator, Code Cloning. Sonargraph* koristi metriku *Structural Debt index.* Svaki od ovih alata poseduje specifičnu bazu znanja.

Drugo pitanje vezano za Cilj 2 glasi: *koja je preciznost (pouzdanost) estimacija tehničkog duga?* Odgovor na ovo pitanje dobijen je testiranjem odabranih projekata u odabranim alatima. Dobijeni rezultati pokazuju da dati alati nemaju istu bazu znanja, odnosno svaki alat koristi zasebna pravila kako bi pronašao probleme u kodu. Sveukupno, *NDepend* je pronašao najviše problema skoro za svaki projekat. Alat koji je dao najveću kumulativnu vremensku estimaciju količine tehničkog duga jeste *NDepend,* u sredini je *SonarCloud,* dok je najmanju kumulativnu vremensku estimaciju dao *Squore.*

Za dva projekta *NDepend* je procenio 5 puta više kumulativne vremenske estimacije TD-a u odnosu na *Squore*, dok je u jednom projektu procenio čak 12 puta veći TD. Može se zaključiti da predstavljeni alati ne daju identičan rezultat, ali bi trebalo da ukažu na deo koda u kojem postoji problem.

7. VALIDNOST ISTRAŽIVANJA

Validnost istraživanja može biti interna i eksterna. Interna validnost se odnosi se na sprovođenje metodoloških koraka, odnosno treba da opiše da li su metodološki koraci adekvatno sprovedeni. U prvom metodološkom koraku, *Definisanje istraživačkih pitanja*, korišćena je *GQM* metodologija koja je veoma korisna za merenje kvaliteta softvera. *Pregled literature*, izvršen pretragom literature na *Google Scholar* pretraživaču. *Pregled postojećih ASAT rešenja* je odrađen detaljno, jer je na osnovu pregleda literature proistekao inicijalni skup od 14 alata, koji je dalje detaljnije proučavan direktnom *Google* pretragom. Odabir 4 alata je usledio na osnovu jasno definisanih kriterijuma. Šest *open-source* projekata je takođe odabrano na osnovu dva jasno definisana kriterijuma. Analiza projekata je izvršena u svakom od odabranih alata uz prethodna podešavanja potrebnih parametara i okruženja.

Eksterna validnost govori o relevantnosti dobijenih zaključaka. Ono što može biti značajno kada se radi o eksternoj validnosti jeste činjenica da su u ovom istraživanju analizirani *open-source* projekti izabrani na osnovu postavljenih kriterijuma. Dati projekti ne mogu predstavljati ceo skup *open-source* projekata. Takođe, nisu testirani projekti iz industrije, odnosno *closed-source* projekti pa se zbog toga ne može govoriti o zaključcima za takve projekte

8. DISKUSIJA

Rezultati ovog istraživanja pokazuju da *ASAT* alati koji su bili predmet istraživanja ne pružaju iste rezultate analize nad istim projektima. Tri od ukupno četiri korišćena alata koriste isti model identifikacije TD-a (*SQALE* metodologiju), ali i pored toga daju različite rezultate analize. To potvrđuje da dati alati imaju različite baze znanja, osnosno skupove pravila koje koriste kako bi identifikovali i estimirali probleme u kodu. Svaki od ovih alata poseduje definisane metrike za estimaciju TD-a.

Neke od ovih metrika se nalaze u većini ovih alata (na primer, metrika *Debt*), ali većina njihovih metrika se ipak razlikuje. Alati *SonarCloud*, *NDepend* i *Squore* su sličniji jedni drugima iz razloga što koriste *SQALE* metodologiju, svi prepoznaju *Code debt*, i daju estimaciju TD-a u vremenu. *Sonargraph* se razlikuje od prethodno navedenih alata iz razloga što ne implementira *SQALE* metodologiju, ne prepoznaje *Code debt* i ne daje estimaciju TD-a u vremenu. Ovaj alat prepoznaje *Architectural debt* i *Design debt*. *NDepend* takođe prepoznaje i *Architectural* i *Design debt.* Drugi istraživači koji su poredili ASAT alate došli su do sličnih zaključaka. Oni potvrđuju da se ASAT alati prilično razlikuju u pogledu osetljivosti na probleme [10].

9. ZAKLJUČAK

U ovom radu opisana je komparativna analiza ASAT alata u svrhu identifikacije i procene TD-a u *.NET* projektima. Alati su poređeni na osnovu rezultata koji su dobijeni analizom šest odabranih projekata u navedenim alatima.

Izveden je zaključak da alati koji su bili predmet analize daju različite rezultate, ali bi svi trebali da ukažu na deo koda u kojem postoji problem. U fokusu ovog istraživanja bili su ASAT alati koji, između ostalih, mogu da analiziraju *.NET* projekte.

U budućim istraživanjima plan je da se analiziraju i projekti napisani u drugim tehnologijama, odnosno programskim jezicima, kao što je *Java*.

10. LITERATURA

[1] A. Figueroa, „ROOTSTRAP,“ [Na mreži]. Available: https://www.rootstrap.com/blog/what-is-technical-debt-with-examples/. [Poslednji pristup 5. 12. 2021.].

[2] N. T. R. R. S. F. N. R. B. P. C. D. C. A. P. G. L. C. I. D. F. C. S. R. S. V. Mandic, „Technical and Nontechnical Prioritization Schema for Technical Debt: Voice of TD-Experienced Practitioners,“ *IEEE SOFTWARE*, 2021..

[3] R. L. N. I. O. P. Kruchten, „Technical Debt: From Metaphor to Theory and Practice,“ *IEEE SOFTWARE*, 2012..

[4] R. B. S. M. A. Z. M. Beller, „Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software,“ *2016.*.

[5] E. M. A. M. R. S. A. T. F. A. Fontana, „An experience report on using code smells detection tools,“ u *Fourth International Conference on Software Testing, Verification and Validation Workshops*, 2011..

[6] M. W. Suman, „A Comparative Study of Software Quality Models,“ *International Journal of Computer Science and Information Technologies*, t. 5, 2014.

[7] C. I. C. H. Isaac Griffith, „An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models,“ 2017.

[8] D. R. C. I. Z. C. A. D. B. W. Isaac Griffith, „The Correspondence between Software Quality Models and Technical Debt Estimation Approaches,“ u *2014 6th IEEE International Workshop on Managing Technical Debt*, 2014.

[9] L. F. R. V. C. T. S. M. R. O. S. Nicolli S. R. Alves, „Towards an Ontology of Terms on Technical Debt,“ 2014.

[10] E. B. Rini van Solingen, The Goal/Question/Metric Method: a practical guide for quality improvement of software development, McGraw-Hill Publishing Company, 1999..

Kratka biografija

Milica Stanković rođena je u Šapcu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Inženjerstvo informacionih sistema odbranila je 2021.god.
kontakt: milicavishnjich@gmail.com