Využitie statickej analýzy kódu pri vývoji softvéru *

Lukáš Častven
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xcastven@stuba.sk

5. november 2021

Abstrakt

Statická analýza je proces, pri ktorom je počítačový kód zanalyzovaný bez samotného spúšťania kódu. Po tejto procedúre sú programátorovi prezentované nájdené chyby, ich možný spôsob opravy a aj varovania o menej závažných nedostatkoch a ich riešenia. Pomocou tejto metódy dokážeme v celom analyzovanom projekte zlepšiť kvalitu kódu a udržať konzistentný štýl, ktorý taktiež spĺňa osvedčené postupy pri vývoji softvéru. Veľkou výhodou je tiež urýchlenie hľadania chýb a softvérových defektov v porovnaní s manuálnou kontrolou. V tomto článku pochopíme, prečo developeri používajú nástroje statickej analýzy, ako ich používajú na opravu a zlepšenie kódu a ako ich implementujú do ich pracovného prostredia.

1 Úvod

S rastom komplexity vyvýjaného softvéru rastie aj jeho minimálna požadovaná kvalita. Chyby a defekty softvéru dokážu firmám spôsobiť nemalé finančné straty. Existujú preto viaceré metódy ako kvalitu softvérových riešení zlepšiť a udržať počas vývoja. Napríklad podrobné testovanie alebo revízie nového kódu iným developerom. V tomto článku si bližšie priblížime metódu statickej analýzy, ktorá je používaná na odhalenie chýb a defektov a na udržovanie kvality kódu na požadovanej úrovni. [1, 2]

O statickej analýze samotnej sa dozvieme v časti 2, presnejšie, pochopíme čo to je (č.2.1) a ako funguje (č.2.2). Ako sa nástroje statickej analýzy používajú v praxi je uvedené v časti 3, kde zistíme ich benefity (č.3.1) a aj nedostatky (č.3.2). Ako sú nástroje implementované v pracovnom prostredí je podrobnejšie vysvetlená v časti 4. Ako s nimi pracujú developeri v priebehu práce zitíme v časti 5. A v neposlednom rade zistíme aké nové funkcionality by chceli vývojari vidieť v budúconsti, časť 6.

2 Princípy statickej analýzy

2.1 Definícia statickej analýzy

Statická analýza je proces, pri ktorom je kód posudzovaný bez spustenia samotného programu a bez vstupu. Preto sa nazýva statická. Analyzuje sa zdrojový a v niektorých prípadoch aj objektový kód[1]. Po úspešnom ohodnotení kódu, sú developerovi zobrazené nájdené chyby a defekty. Taktiež sú poskytnuté vysvetlenia, prečo sú najdené chyby reálnymi chybami a ako ich je možné opraviť. [3]

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2021/22, vedenie: Ing. Zuzana Špitálová
[1]Väčšinou ide o súbory s rozšírením nazvu „.o".

2.2 Funkcia statickej analýzy

Existuje veľa typov nástrojov statickej analýzy, ale nie všetky poskytujú rovnaké funkcionality ako iné. Našťastie hlavný postup je jasne definovaný. Skladá sa len z dvoch krokov:

1. Analýza zdrojového (objektového) kódu.
2. Informovanie o chybách, defektoch a narušení osvedčených postupov.

Aby sme lepšie pochopili, čo statická analýza robí, pozrieme sa na konkrétny nástroj FindBugs. Tento nástroj sa používa na odhalenie chýb v programovacom jazyku Java. [4]

Krok (1) je za pomoci FindBugs realizovateľný buď cez plugin v integrovaných vývojarských prostrediach, alebo cez príkazový riadok. Po zanalyzovaní kódu, FindBugs rozdelí defekty do kategórií podľa ich závažnosti. Tie sú: vysoká, stredná a nízka. Následovne, ako súčasť kroku (2), poskytne vývojarovi aj niekoľko možných návrhov rýchlych opráv[2]. [1, 5]

3 Využitie pri vývoji softvéru

V praxi nie je využitie statickej analýzy bezproblémové. Samozrejme, benefity tejto metódy sú jasne viditeľne, avšak developerov viac odrádzajú ťažkosti a nedostatky ako pozitívá, ktoré táto metóda poskytuje. V tejto časti si rozoberieme vedeckú prácu **Why don't software developers use static analysis tools to find bugs?** [1] a zistíme prečo je predošlé tvrdenie pravdivé.

V tomto príspevku autori uskutočnili rozhovory s dvadsiatimi vývojarmi na tému statická analýza. Zozbierané odpovede zkategorizovali do niekoľkých okruhov a rozdelili na pozitívne a negatívne.

3.1 Benefity

Automatické hľadanie chýb

5 z 20 účastníkov tejto práce si myslí, že nástroje statickej analýzy sa oplatí používať práve z tohto dôvodu. Manuálna kontrola je zdĺhavá a namáhavá, oproti automatickej statickej analýze.

„Hoci čo, čo zautomatizuje nudnú prácu je skvele."

Pred-integrované v prostredí

Ako dôvod prečo využívajú statickú analýzu, zvolili 3 účastníci pred-integráciu v ich pracovnom prostredí. Veľa integrovaných vývojových prostredí už má v sebe zabudované tieto nástroje, aj keď ide iba o ľahké verzie, ktoré nevedia odhaliť všetky chyby.

Udržiavanie tímových praktík

Zvýšenie povedomia o možných problémoch, alebo o chybách z nepozornosti skôr vo vývojovom cykle je kľúčové ku udržaniu tímoveho vývojového úsilia. Taktiež sa ľahšie udržuje jednotný štýl kódu. 7 z 20 developerov označili tento benefit ako rozhodujúci v otázke prečo používajú statickú analýzu.

Komunikácia

Nástroje statickej analýzy sú užitičné pri komunikácií a pri presadzovaní štandardov a štýlov vo vývojovom tíme, uviedli 2 respondenti.

[2] ang. Quick fixes

Nastaviteľnosť

3 účastníci využívajú nástroje statickej analýzy pre ich schopnosť vytvorenia vlastných typov chýb, čo im umožňuje automatické detekovanie chýb, ktoré sú špecifické pre daný projekt.

3.2 Nedostatky

Výsledok analýzy

Až 14 z 20 respondentov označilo zlý výstup a úbohú prezentáciu ako dôvod, ktorý ich odrádil od určitého nástroja statickej analýzy.

Je známe, že tieto nástroje produkujú aj falošné pozitíva, čiže hlásenia o chybách, ktoré nie sú naozajsnými chybami. Ak nástroj produkuje viac falošných pozitív ako reálnych pozitív, uživateľ musí manualné prefiltrovať všetky a zistiť, čo je správne. Toto daného uživateľa demotivuje, čo spôsobí vymenenie nástroja, alebo v horšom prípade, celkové vynechanie statickej analýzy.

V masívnejších projektoch je počet defektov veľký a ak prezentácia je neprehľadná a nečitateľná, developer sa ľahko stratí v kvantách informácií. To znamená, že viac času je minutého na opätovné zorientovanie sa, ako na samotné opravenie chýb a toto, ako v predošlom odseku, môže viesť k vynechaniu statickej analýzy kompletne.

Tímová práca

Vývoj softvéru je prevažne tímová práca, preto je dôležité, aby inžinieri nástrojov statickej anaýzy brali tento dôležitý fakt do úvahy pri ich vytváraní. Jeden vývojar uviedol, že síce sú tieto nástroje užitočné, ale neexistuje jednoduchý spôsob ako zdielať nastavenia v tíme. Z tohto úkonu sa takto stáva nepraktický manuálny proces nastavovania, ktorý v prípade zmeny štandardov a nastavení zapríčiní zmätok a chaos.

Viacerí respondenti by chceli aby tieto nástroje zjednodušili komunikáciu a kolaboráciu v tíme. Aby nemuseli narušiť svoj pracovný priebeh.

Spolu označilo túto kategóriu 9 z 20 opýtaných ako závažný problém.

Netriviálna nastaviteľnosť

Každý projekt je inakší, ma iné požiadavky, restrikcie a kvality. Preto je nastaviteľnosť dôležitý aspekt pre využívaný nástroj. Ale ako uviedlo 17 z 20 developerov, táto vlastnosť je silne netriviálna alebo nespĺňa vývojarove predstavy.

Vývojar potrebuje niekoľko hodín len na to aby sa naučil ako nastaviť daný nástroj. *„Mnohe nástroje sú tak ťažko nastaviteľné, že vám v tom rovno zabránia."*

Problém, ktorý mala väčšina respondentov, je nemožnosť dočasne vypnúť upozornenia na špecifický typ defektov. Niektoré nástroje poskytuju vypnutie natrvalo, ale toto sa zas nepáči developerom, lebo nikto nevie vopred povedať, či daná, teraz ignorovateľná chyba, bude taká aj v budúcnosti.

Nejasnosť

Väčšina developerov nedostatočne využíva túto metódu pre nejasnosť vo výstupe. Neschopnosť porozumieť nájdeným chybám je pre 19 z 20 vývojarov bariera. Nezrozumiteľné a nevyužiteľné vysvetlenia alebo málo poskytnutých informácií sú hlavné faktory vzniku nejasností.

Najčastejšie spomínaný problém je nedostačujúca alebo neefektívna implementácia rýchlych návrhov opráv. Väčšina z respondentov by prijala návrhy počas opravovania nájdených chýb. *„Ak mi dokážeš povedať o chybe, tak by si mi mal vedieť aj povedať ako ju opraviť."*, vyhlásil jeden z respondentov.

4 Implementácia nástrojov statickej analýzy

Existuje viacero typov nástrojov statickej analýzy. Nástroje implementované v integrovaných vývojových prostrediach (IDE), ktoré bežia v pozadí, poskytujú instantú spätnú väzbu počas práce, avšak tento typ nachádza len jednoduché chyby nie komputačne náročné na odhalenie.

Niektorí developeri nepoužívajú IDE, vtedy prichádza do úvahy nástroj integrovaný do kompilátora daného programovacieho jazyka.

A v neposlednom rade poznáme aj rigorózne analyzátory, ktoré sa väčšinou spúšťajú cez noc, lebo dlho a podrobne analyzujú daný kód. [1, 2]

5 Využitie v praxi

Samozrejme dôležitá je prax, to ako uživateľ používa a vníma daný nástroj [2]. Diagram Obr. 1 znázorňuje ako sú nástroje statickej analýzy používané v praxi.

Kedy sú používané

Developeri využívajú tieto nástroje behom voľna (v časoch medzi stretnutiami, alebo keď majú časové okno). Opravovanie nájdených chýb je v priemere vykonávané v úsekoch od 10 do 30 minút a oprava jedného nálezu zaberie zhruba menej ako 1 deň.

Ciele pri používaní

Zvyčajne je vývojarovým cieľom opraviť všetky defekty alebo ich opraviť čo najviac v stanovenom časovom úseku. Preto je čas hlavný dôvod ukončenia opravy a toto priamo ovplyvňuje aké chyby si developer vyberie na opravu.

Výber chýb na opravu

Developer si prevažne vyberie chybu, ktorú vie opraviť, má na to znalosti o celkovom projekte alebo má skúsenosť s daným nástrojom statickej analýzy. Pri varovaniach na nedostatky, ktoré ale nie sú chyby, však dominuje skúsenosť s nástrojom. Preto je dôležité aby dizajn daného nástroja podporoval správanie, ktoré vedie k oprave, aj keď vývojar nedokáže chybu opraviť.

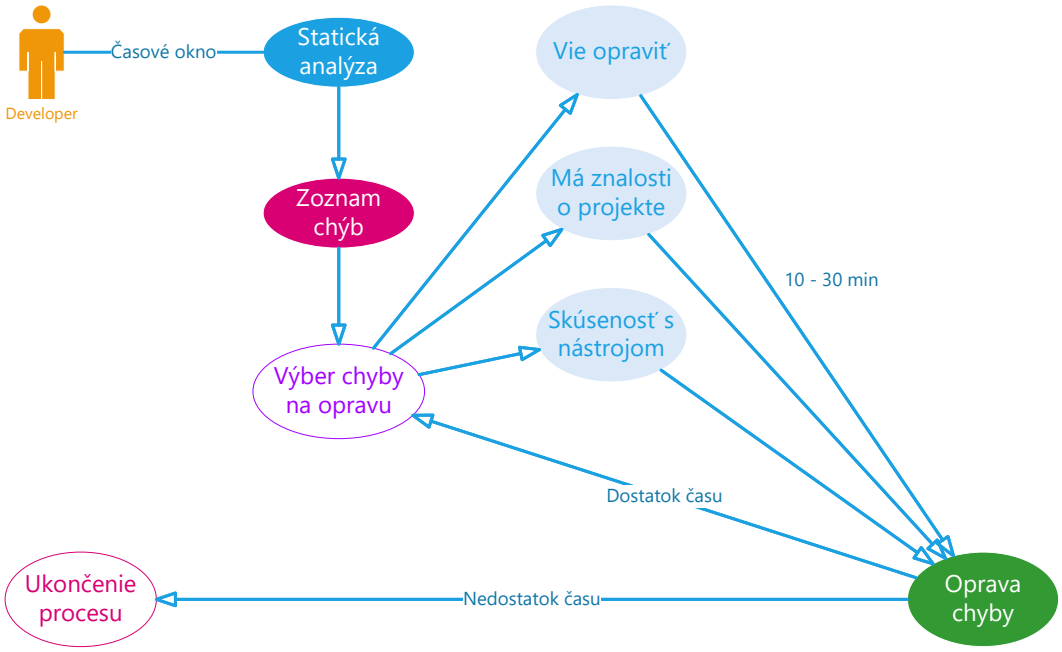
6 Návrhy na zlepšenie

Najvyužívanejšie funkcionality sa týkajú opravy chýb. Návrhy rýchlych opráv by mali ukázať porovnanie terajšieho kódu a kódu po aplikovaní návrhu. Taktiež by sa developerom páčila možnosť aplikovania opravy rovno z nástroja. Táto aplikácia by mala mať viacero možností, aplikovanie celého návrhu, iba časti, alebo vôbec.

Samotné zobrazenie chýb má tiež zopár aspektov na vylepšenie. Malo by byť čo najrýchlejšie, aby nenarušovalo pracovný priebeh, čiže najlepšie vyhodnocované v pozadí s okamžitou spätnou väzbou. A malo by aj uľahčiť kolaboráciu a komunikáciu. Napríklad posielanie správ alebo zdielanie nálezou s inými.

Nakoniec developeri by preferovali aj iné formy výstupu ako len list s chybami. Grafy, diagramy alebo teplotné mapy s modulmi kódu ako nódami. [1, 2]

Obr. 1: Práca so statickou analýzou v praxi.



7 Záver

Statická analýza pomáha developerom zlepšiť a udržať kvalitu kódu v projektoch. Dokáže nájsť chyby urobené z nepozornosti, bezpečnostné defekty, nedodržiavanie osvedčených postupov, narušenie štýlu a mnohé iné nedostatky.

Aj keď v teórii prináša statická analýza veľa pozitív, v praxi nie je práca s nástrojmi statickej analýzy bezproblémová, to zapríčiňuje nedostatočné využívanie. Chyby v implementácií tejto metódy do pracovných prostredí vývojarov prevážia jej benefity.

Preto by mali inžinieri statickej analýzy viac počúvať spatnú väzbu k ich nástroju od uživateľov a rozvíjať funkcionality v tomto smere.

Reakcia na prednášky

Spoločenské súvislosti. Každým dňom sa naša spoločnosť spolieha viac a viac na softvérové riešenia, či už v medicíne pri záchrane životov, vo finančnom sektore a napríklad aj v doprave. To zapríčiňuje rast minimálnej kvality vyvýjaného softvéru, lebo aj tá najmenšia chyba, napríklad v medicíne, mohla spôsobiť smrť. Preto veľa IT tímov používa nástroje statickej analýzy. Tie dajú developerovi viac času na riešenie dôležitých chýb a defektov, čím sa zdvíha úroveň finálneho softvérového riešenia.

Historické súvislosti. V sedemdesiatych rokoch minulého storočia boli nízko levelové jazyky široko rozšírené. Avšak v týchto jazykoch bolo veľmi ľahké urobiť chybu z nepozornosti. Toto motivovalo Stephena

Johnsona pri výrobe prvého nástroja statickej analýzy. Navrhol ho počas jeho práce v Bell Labs a nazval ho Lint. Tento nástroj, spolu aj s postupom ako rozdeliť vývoj na čisto programovanie a potom na retroaktívne hľadanie chýb s pomocou Lintu. [6]

Technológia a ľudia. Vďaka statickej analýze je kvalita kódu počas celého vývojového cyklu udržovaná na predom stanovenej úrovni. Toto je zapríčinené viacerými benefitami, ako napríklad automatické hľadanie chýb a defektov a udržovanie konzistentného štýlu kódu v celom projekte. Avšak implementácie nástrojov nie sú dokonalé a majú určité nedostatky, ktoré dokážu uživateľa demotivovať a to môže zapríčiniť vynechanie statickej analýzy počas vývojového cyklu. Našťastie, inžinieri týchto nástrojov dávajú veľký dôraz na spätnú väzbu od developerov a na jej základe sa snažia napraviť a vylepšiť ich nástroje. A tým celkovo posúvaju obor statickej analýzy vpred.

Udržateľnosť a etika. Udržateľnosť je jedna z najdôležitejších vlastností dlhodobých IT projektov. Statická analýza priamo podporuje udržatelnosť softvérových riešení, tým že poskytuje IT tímom mnohé benefity, medzi ktoré patria napríklad automatické hľadanie chýb alebo dodržiavanie štandardných štýlov kódu v celom projekte. A taktiež napomáha developerovi pri dodržovaní určitých bodov etického kódexu (Software Engineering Code of Ethics and Professional Practice).

Literatúra

[1] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?," IEEE, may 2013.

[2] L. N. Q. Do, J. Wright, and K. Ali, "Why do software developers use static analysis tools? a user-centered study of developer needs and motivations," 2020.

[3] Wikipedia, "Static program analysis — Wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Static%20program%20analysis&oldid=1050273451`, 2021. [Online; accessed 02-November-2021].

[4] "Findbugs." `http://findbugs.sourceforge.net/`. accessed 02-November-2021.

[5] N. Ayewah, W. Pugh, D. Hovemeyer, J. D. Morgenthaler, and J. Penix, "Using static analysis to find bugs," vol. 25, pp. 22–29, Sept. 2008.

[6] S. C. Johnson, "Lint, a c program checker," in *COMP. SCI. TECH. REP*, pp. 78–1273, 1978.