Bírálat
Dr. Ferenc Rudolf
„Improved Bug Prediction Through Conceptual Metrics and Machine Learning/ Továbbfejlesztett szoftverhiba előrejelzésfogalmi metrikák és gépi tanulás segítségével”

című MTA Doktora címre benyújtott értekezéséről

Szoftverrendszerek, nagy programok hibamentessége elméletileg bizonyíthatatlan, ezért a hibák előrejelzése, kiszűrésére, valószínűsítése elengedhetetlen a szolgáltatások minőségének biztosításában. Minden előrejelzéssel megtalált hiba egy potenciálisan nagyobb kárt eredményező meghibásodást véd ki. A dolgozat témája igen időszerű, eredményei a hibák előrejelzéséhez új módszerek kidolgozásával járulnak hozzá.

A szoftver az elmúlt 75-80 év során jelent meg az emberi mesterséges alkotások világában, és napjainkban a mindennapi élet egyre szélesebb területén megkerülhetetlenné vált használata. Olyan új tulajdonságokkal rendelkezik, ami megkülönbözteti minden más mesterséges emberi alkotástól.

Első megjelenési formája: akár papírra leírt szöveg lehet, újabban számítógépes szövegszerkesztővel készített szövegnek tekinthető. Digitális formába kódolva egy, vagy több szoftver alkalmazásával számítógépen futtatható programkód készül belőle. Ennek futtatásával keletkező eredmények segítenek a felhasználási cél elérésében. A szoftver digitális kódja tetszőleges sok példányban elkészíthető, felhasználása sokszorozható. A használattól nem hibásodhat meg, de lehet benne hiba.

Ezzel az igen leegyszerűsített jellemzéssel a szoftverminőség kérdésére irányuló módszerek, a minőségi mértékek különleges nehézségének magyarázatát kívántam előkészíteni.

Az 1970-es évek elején az MTA CDC-3300 gépén még elég jó sikerrel tudtam néhány száz soros FORTRAN és COBOL programok a forráskódjában hibát keresni. Ugyanez ma a százezer soros, vagy még nagyobb programok esetében nem lenne járható út.

A szoftver minősége szorosan következik magából a forráskódból, azonban az alkalmazásban várható minőséget nehéz közvetlenül kimutatni. A sok fázisból álló tesztelést követő üzembe helyezés után már a megvalósult, felhasználási minőség jelenik meg.

A forráskód mindent hordoz, amitől a szoftver minősége függ, ezért fontos annak a vizsgálata, hogy a forráskódból önmagában milyen minőségi jegyekkel jellemezhető, és különösen milyen hiba lehetőségek, veszélyek prognosztizálhatók. Általában a statikus elemzők a forráskódnak csak azt a részét vizsgálják, amit a fordítóprogram felhasznál. A programozási nyelv forráskódjára alapozva számos metrika, statisztika, elemzési módszer épült ki. Többek között az értekezésben említett programszeletelésre épülő COLUMBUS rendszer, amelynek kifejlesztésében a jelölt is részt vett, és újabb, bővített változata az Open Static Analyzer nagyon sokrétű statikus elemzést szolgáltat.

A szoftverek forráskódja azonban olyan elemeket és névválasztási lehetőségeket is tartalmaz, amelyek a végrehajtást nem befolyásolják, mégis lényeges mutatói lehetnek a szoftver minőségének. Az egyik ilyen nyilvánvaló lehetőség a kommentek használata. A másik, kevésbé nyilvánvaló, a nevek (változók, objektumok, metódusok, címkék, stb,) választása. Ezek mind olyan lehetőségek, amelyek a programkód humán megértéséhez járulnak hozzá, de nem befolyásolják a kód logikai működését. A szoftverfejlesztés minőségi elvárásai, amihez szabványok is készültek, elvárásokat tartalmaznak a szoftver dokumentáltságára és fenntarthatóságára, nagy mértékben ezen a két összetevőn múlik. Ferenc Rudolf értekezésének első részében azokat az eredményeit mutatja be, amelyek a teljes forráskódnak, mint szövegnek nyelvtechnológiai elemzésére alapulnak. A hagyományos szerkezeti metrikák, mint a kohézió és csatolás fogalmi változatait szerőztársaival dolgozta ki, és igen szigorú empirikus kiértékeléssel mutatta ki használhatóságukat. A második részben a mesterséges intelligencia legújabb eszköztárának, mélytanulásnak használatát az egyre nagyobb nyílt szoftverkód-készletekre és hibagyűjteményekre vizsgálja érdemi új eredmények kimutatásával.
Az értekezés tartalmi összefoglalója:
Az irodalomjegyzék nélkül 123, összesen 144 oldalas dolgozat 8 fejezetet és két függeléket tartalmaz.
Az első fejezet a témakör általános bevetője, és egy-egy bekezdésben leírja az 5 tézispontot is, amelyeket a 3-7. fejezetek tartalmaznak.
A második fejezetben a felhasznált alapvető fogalmakat és eljárásokat ismerteti, többek között a statikus forráskódelemzés, a látens szemantikai indexelés, statisztikák, gépi tanulás területéről.
A 3-7. fejezetekben egységes szerkezetben fejti ki a tézispontokat: az irodalmi háttér és előzmények, az új, elvi modell kifejtése, a konkrét szoftvergyűjteményeken végzett elemzés, az értékelés, a saját szerep és a kapcsolódó saját cikkek sorrendben.
A I. rész a fogalmi kohézió és csatolás három tételkörét tartalmazza.
Az Objektum Orientált (OO) programozási paradigma a nagy szoftverek dekomponálásával kezelhetőbb bonyolultságú építkezést vezetett be. A komponensek, azaz objektumok belső felépítését és egymáshoz kapcsolódását a kohézió erősségével és a csatolás gyengeségével jellemzik. A dolgozat feltételezése, hogy a forráskód strukturális része mellett, amit a fordítóprogram használ, a humán megértést támogató strukturálatlan részek (kommentek és nevek) szintén valamilyen mértékben hordozzák a szoftver feladatának és megoldásának tartományát és kiszámítási logikájának lényegét. Ebből kiindulva a forráskód teljes szövegét felhasználva a nyelvtechnológiák módszereire építve új kohéziós és csatolási mutatókat lehet keresni. A következő három fejezetben egy-egy ilyen új mutató definiálása után a strukturális mutatókkal való részletes összehasonlítást végzi el.
A 3. fejezetben kezdi meg a konkrét kohéziós mérőszám definiálását. Az OO rendszerhez gráfreprezentációt vezet be, ami az egyes osztályokon belül a metódusokat súlyozott élekkel köti össze. A súlyok a metódusok közötti hasonlóságot mérik. A hasonlóságot a metódusok szövegeinek koherenciája adja. Ehhez a forráskódot nyelvfeldolgozáshoz alkalmassá kell alakítani. A módszerhez kifejlesztették az IRC^{3}M eszközt, ami a szövegkorpuszt állítja elő, abból az LSI indexekre épülő szemantikus teret, majd a

metódusok közötti hasonlóságot, amit a vektorjaik közötti cosinus érték mér. (Ez a módszer a 90-es években az információ-visszakereső rendszerekhez fejlődött ki) Ezeknek az átlaga (ha pozitív) adja az osztály fogalmi kohéziójának mértkét. (C3 metrika.) Az így definiált fogalmi kohéziós mértéket több nyilvános szoftvergyűjteményen összehasonlítják más strukturális kohéziós mértékekkel. Két kutatási kérdést tűznek ki, $RQ_{3.1}$ és $RQ_{3.2.}$ Igazolódik a várakozás, az $RQ_{3.1}$, hogy a kohézió jellemzéséhez a C3 metrika új dimenziót ad. A másik esettanulmányban az osztályok hibára való hajlamosságának előrejelzésében is jó eredményt adott C3 kombinálása más metrikákkal. Fontos itt megemlíteni, ami a későbbi esettanulmányokra is igaz, hogy a kiértékeléshez a csoport saját Columbus rendszerét használták és bővítettek az új metrika algoritmusaival. A jelöltnek az esettanulmányok megszervezése, és jelentős részének kivitelezése és kiértékelése volt fő hozzájárulása. A kapcsolódó cikkre (Tézisfüzet [6]) 202 független hivatkozás történt

A 4.rész a szoftvermódosítások kihatásainak előrejelzését támogató csatolási mérőszám fogalmi változatával foglalkozik. A kiindulás az előző fejezetben bevezetett metódusok közötti fogalmi összehasonlítás, azonban itt a különböző osztályhoz tartozó metódusok hasonlóságát kell használni. Az csatolási mérték felépítése átlagokból: egy osztály metódusa-egy másik osztály összes metódusa, egy osztály összes metódusa és egy másik osztály összes metódusa, végül egy osztály metódusai és a többi osztály metódusai adják a metódus-osztály, az osztály-osztály és végül az osztály-többi osztály csatolási mértékét. A legfinomabb csatolási metrika, $CCBC_m$, az osztályok között a leghasonlóbb metóduspárhoz tartozó maximum értéket használja. A nagy csatolási értékek a módosítások kihatását valószínűsítik. A metrikák bemutatása után következik a fejezet érdemi része, az $RQ_{4.1}$ kutatási kérdésre való válasz megadása.

A Mozilla rendszer forráskódját (4800 osztály, 740000 kódsor, 1021 hiba, a javító patch-fájlokkal) használták, ami lehetővé tette a javítások kiterjedésének pontos követését. Ennek kigyűjtése után a fogalmi és 9 szerkezeti csatolási mértéket összehasonlították a javítások kiterjedésének előrejelzésében. Az igen körültekintő kiértékelés a $CCBC_m$ fogalmi maximum-csatolási metrikát az összes többinél hatékonyabbnak mutatta, ami az $RQ_{4.1}$ kutatási kérdésre pozitív választ adott. Fontos megjegyzés a 46 oldal első mondata, hogy a fogalmi metrika csak akkor működik, ha betartják az elnevezési konvenciókat a kommentekben és változókban.

A fogalmi metrika és kiszámításának implementálása a hivatkozott közös dolgozat társszerzőinek eredménye, a strukturális metrikákkal való összehasonlítás, a Mozilla forráskód és Bugzilla hibafájl előkészítése, az összehasonlítások szervezése, a kiértékelések többségének végrehajtása a jelölt eredménye. A kapcsolódó cikkre (Tézisfüzet [7]) 144 független hivatkozás történt

A fogalmi kohéziós és csatolási metrikákkal foglakozó harmadik, 5. számú fejezetben Ferenc Rudolf egy új finomítást dolgozott ki. Az átlagértékekkel foglalkozó metrikák helyett csak az adott t küszöbéréket meghaladó metódushasonlóságokat, illetve osztályhasonlóságokat vette figyelembe. Az ilyen párokhoz az 1 élsúlyt rendelte. Ezzel a 3. fejezetben az osztály metódusai között bevezetett élsúlyozott teljes gráf helyett egy ritkább, súlyozatlan élekből álló gráf keletkezik. Ebben az osztályon belüli metódusok gráfjában az összefüggő komponensek száma méri a kohézió hiányát. Egy osztály paraméteres csatolási mérőszáma az osztályok közötti gráfban a vele összeköttetésben lévő osztályok száma.

Az így bevezetett új paraméteres metrikára 4 kutatási kérdést tűz ki, és az előző fejezetekhez hasonló alapossággal végzi el az empirikus kiértékeléseket. A Mozilla szoftvergyűjteményből a szövegkorpuszt két változatban, szőtövezéssel és anélkül készítették el. A kohézió hiányát mérő új metrika a főkomponens analízisre épülő összehasonlításban hatékonynak bizonyult. Közel áll több strukturális metrikához, egyszerű kiszámíthatósága miatt viszont velük szemben előnyös. Kombinálása strukturális metrikákkal valamivel jobban javít az előrejelzésen mint a C+ metrika. Statisztikai összehasonlítást végeztek több MI technikát használva az új kohéziós metrikához a hibára való hajlamosság előrejelzésére, összehasonlítva a két szövegelőkészítési változatot. A szótövezés kicsit javít a módszerek jóságán. A fejezet döntő részben a jelölt eredményeit tartalmazza. A munkára 50 független hivatkozás történt.

1. Kérdés: A fogalmi alapú kohéziós és csatolási metrikák a szerkezeti metrikákhoz hasonlóan jól mutatják a szoftver fontos minőségi tulajdonságait, mint például a hibára való hajlamosságot, a módosítások várható kiterjedését. A dolgozat esettanulmányai a kész szoftverek történeti adatain ellenőrzik a metrikák használatát. Kérdés, hogy a tervezés során nem lehetne-e a folyamatosan épülő forráskódon monitorozni a fogalmi metrikákat és ezzel a jó mutatók felé alakítani a tervet?

A II. rész témája a hibaelőrejelzés gépi tanulása.
Első fejezete, a 6.Fejezet a tanulás és az előre jelző módszerek teszteléséhez elengedhetetlenül szükséges jó minőségű hiba-gyűjtemény építését ismerteti. Egy igen nagy célkitűzés sikeres megvalósításának részleteit láthatjuk ebben a fejezetben
A különböző hiba-adatgyűjtemények tartalmazzák a teljes forráskódot, a hibát tartalmazó forráskód részt, a teljes forráskód kiszámított metrikáit és sok egyéb kapcsolódó információt. Alapos előkészítő irodalom és jellemzők elemzése után a kiválasztott 5 nagy gyűjteményen szűrések és egységesítés után készült az egységesített hibagyűjtemény. Az egyes gyűjtemények metrikáit is megtartották, mellette az OSA rendszer metrikáit is kiértékelték. A felépített gyűjtemény hibaelőrejelzésben való használatát is tesztelték, sokféle tanítóminta-választással, a mintákra egymás közötti, vagy a teljes gyűjteményre, és az egészet az eredményekkel együtt nyilvánosan hozzáférhetővé tették. Az így létrejött adatbázis 47618 osztály szintű, továbbá 43744 fájl granularitású adatot tartalmaz. A feldolgozások részleteit az Appendicesben található táblázatok tartalmazzák, amelyek a hibagyűjtemény használói számára fontos részleteket tartalmaznak. Az eredmény egy minden eddiginél nagyobb, és egységesen kezelhető hibagyűjtemény, az eredeti gyűjteményeket saját és az OSA metódusaival kiértékelve. Ezzel a szoftverminőség és hibakeresés eszközrendszere egy hatékony eszköztesztelő kiegészítést kapott. Az ismertető publikációra (Tézisfüzet [4]) már 14 független hivatkozás történt 2020 óta, és többen használták is publikációjukhoz. A módszertani rész közös kidolgozása és az empirikus vizsgálat az egységesített hiba adatbázis segítségével előállított gépi tanulási modellek szoftverhiba-előrejelző képességéről a jelölt fő hozzájárulásai.
A 7. fejezet részletesen ismerteti az előzőekben felépített egységesített hibagyűjteményen a metrikák alapján a mélyhálós tanulásnak az osztályok hibára hajlamos, vagy nem hajlamos-kategóriába sorolására való használatát. A mintaszerűen felépített tesztelés a mélyháló paramétereinek legkedvezőbb választásával kezdődik. Ezt követi a leggyakrabban használt osztályozási módszerekkel való összehasonlítás. A legjobb eredményt a mélyhálós tanulás és a véletlen erdő módszerek adták, közel azonos F-metrikával. A két módszer együttes használatával egy kicsit még ezen is javítani lehet. Ellenőrizték, hogy az egyes módszerek mennyire érzékenyek az adatállomány méretére. Fontos megállapítás, hogy legjobban a mélyhálós tanulás javul a méretek növekedésével. A mély neuronhálós modell kidolgozása a jelölt eredménye, a többi

közös eredmény. Hasonló kísérletek elvégzésére alkalmas nyílt szoftverrendszert is készítettek. Az eredményeket tartalmazó 2020 nyarán megjelent cikkre (Tézisfüzet [2]) már 11 külföldi független hivatkozás történt.

2. Kérdés. A mesterséges intelligencia legújabb, nagy nyelvi modell alapú lehetőségei használhatók lesznek-e ezen a területen?

A 8. fejezet egyetlen oldalon igen lényegre törően foglalja össze az értekezés eredményeit.
Az első függelék az egységes hibagyűjtemény felépítésének táblázatait tartalmazza. A második függelékben a dolgozatban kidolgozott valamennyi eljárás és a hibagyűjtemény nyilvános hozzáféréseit adja meg.
A dolgozat felépítése igen gondos, az elemzések és a kiértékelések a kapcsolódó publikációkban a teljes megismételhetőség kritériumának megfelelően szerepelnek. A fejezetek végén adja meg a szerző saját, felhasznált dolgozatait. Pontosan elkülöníti a csoportmunkában saját részét. Mind az öt téziscsoportban meghatározó szerepet töltött be.
Összegzés.
Mind az öt tételkör igen fontos gyakorlati jelentőséggel bíró eredménnyel járul hozzá a korszerű szoftvertechnológiákhoz. Az első rész új lehetőségeket mutat meg a fogalmi metrikák használatában, új saját metrikát is bevezetve, kibővíti a szerkezeti metrikákra épülő eszköztárat. A valós, nagy méretű forráskódkészleten végzett kiértékelések sokoldalú összehasonlítással támasztják alá az új metrikák használati lehetőségeit.
A második rész előremutató példája annak, ahogy az egyre nagyobb mennyiségben felgyülemlő adatok kihasználásához milyen új megközelítéssel kell hozzáállni. A különálló szoftverrendszerek hibagyűjteményeinek együttes kezelése fontos módszertani kihívást jelent, amit az egységes hibaadattár felépítése, sokrétű belső feldolgozással kiegészítve úttörő jelleggel valósít meg. A záró fejezetben újdonságnak számít a statikus szoftvermetrikákra épülő hibaelőrejelzés mélyhálós tanulással, és ennek összehasonlító vizsgálata más tanuló módszerekkel. Az eredmények előre jelzik a mélyhálós tanulásnak a növekvő forráskód és hibakészletek feldolgozásában növekvő szerepét. A jelölt vezető szerepe a szerzőtársakkal közös eredmények elérésében egyértelműn igazolt.
Külön érték a publikussá tett hibaadatbázis és a használatához kapcsolódó szolgáltató rendszer, amihez az elérési linkek a második függelékben találhatók. Ez azt igazolja, hogy jelentős szoftver-alkotás is húzódik a bemutatott eredmények mögött.
Az angol nyelven írt dolgozat igen jó stílusú, hibát nem észleltem benne. A magyar nyelvű tézisfüzet elég részletes, pontos összefoglalója az eredményeknek.
Mind az öt téziscsoportot elfogadom, a jelölt részvétele és hozzájárulása meghatározó volt mindegyikben. A dolgozatot nyilvános vitára alkalmasnak tartom.

Budapest, 2024.05.20.

Benczúr András

Dr. Benczúr András
Prof. emeritus
ELTE IK