Bírálói vélemény

Ferenc Rudolf
Improved Bug Prediction Through Conceptual Metrics and Machine Learning

MTA Doktori értekezéséről

Ferenc Rudolf *Improved Bug Prediction Through Conceptual Metrics and Machine Learning* címmel nyújtott be MTA doktori értekezést.

Formai szempontok

Az értekezés angol nyelven íródott, a függelékeken kívül 106 oldalon. Ezt a részletes mérési adatokat tartalmazó első és az online elérhető adatokra hivatkozó második függelékek egészíti ki. A dolgozat 248 elemből álló irodalomjegyzékkel fejeződik be, melyből 27 a szerző (társszerzőként írt) cikkeire hivatkozik.

A dolgozatot a szerző egy rövid bevezető fejezet és a technikai háttér (2. fejezetbeli) összefoglalása után két fő részre osztotta: a 3-5 fejezetek a fogalmi (conceptual) metrika definiálásával és annak alkalmazásaival, míg a 6-7 fejezetek a gépi tanulás hiba előjelzésre való alkalmazásával foglalkozik. Ezek a fejezetek egy-egy tézist (T1-T5) tartalmaznak. A dolgozat végén egy rövid összefoglalót találunk.

A dolgozat felépítése logikus, csak elvétve találunk pontatlan megfogalmazásokat (pl. 7. oldal) vagy olyan részt, mely korábban nem definiált fogalomra hivatkozna. Az angol szöveg könnyen olvasható, a (nem angol anyanyelvű) bíráló kevés nyelvtani hibát vagy elírást talált. Az értekezés számos, jól áttekinthető táblázatot tartalmaz, a dolgozat második felében színek támogatják az egyes értékek kiemelését, ez segítene a dolgozat első felében is. Az alkalmazott grafikonok jól támogatják a dolgozat megértését. A 2. függelékben közölt ismertetett online adatok elérhetőek voltak a bírálat írásakor.

A magyar nyelvű tézisfüzet logikusan felépített, és jól foglalja össze a dolgozat eredményeit és az alkalmazott módszereket.

Tartalmi szempontok

Az értekezés két nagy témakört jár körül, mindkettő a szoftverminőség és karbantartás témakörét érinti. Nagy szoftverrendszerek a teljes életciklusra kivetett költségeinek döntő hányadát a karbantartási költségek és a szoftver minőségének fenntartása emészti fel. Így a szoftverminőség kérdése, különösen a hibák potenciális biztonsági következményeit is figyelembe véve, továbbra is kritikus probléma a szoftveriparban. Ennek megfelelően a disszertáció témája aktuális, melyet a szerző az irodalom alapos ismerete mellett korszerű eszközökkel dolgozott fel.

Az első témakör egy új, nem strukturális jellegű szoftvermetrika, az ún. fogalmi (conceptual) kohéziós metrika bevezetése és alkalmazása a hiba előrejelzésre és hatásvizsgálatra vonatkozóan. A fogalmi metrika a szoftverekben lévő, de a forráskódból explicit nem megtalálható információkat, kapcsolatokat próbálja hasznosítani az adott szoftveregység (pl. osztály) kohéziója mérésére.

Ilyenek az alkalmazott kommentek, változónevek melyekről feltételezhető, hogy valamilyen módon tükrözik az osztály fogalmi rendszerét. Bár az ilyen módszerek hátránya, hogy sem a konzisztens névhasználat sem pedig a kommentek up-to-date jellege sem garantált, az irodalomban találunk ilyen módszereket. A szerző bemutatja, hogy az általa definiált metrika egyrészt független a strukturális metrikáktól és azokat kiegészítve jó hibapredikciós jelleggel bír. Itt hiányoltam az utalást arra nézve, hogy egyéb, nem a forráskódban tárolt információk használata felmerült-e. Elsősorban a verziókezelő rendszerekben tárolt egzakt, jól kinyerhető adatokra gondolnék, hiszen két (vagy több) osztály implicit kapcsolatait (csatolását) gyakran jól mutatja, ha azonos commit-okban módosítják őket.

A fogalmi kohéziós metrika kiszámításához alapul vett egység a függvények/metódusok (17. oldal). Számos objektumelvű szoftverben találkozunk triviális getter/setter metódusokkal, melyek akárcsak egy-két utasításból állnak. Ezek figyelembevétele torzíthatja a metrika értékét (a 3.5.1 rész említi is ezt a problémát).

A C++ nyelvben az egyes osztályok interfészéhez gyakran tartoznak globális függvények, sőt bizonyos esetekben ez az ajánlott vagy egyáltalán lehetséges megoldás (pl. tipikusan ilyen a kiíró/beolvasó vagy a relációs operátorok definiálása). Nemegyszer pont ezek a műveletek kapcsolnak össze több osztályt, pl. a `std::getline(std::istream&,std::string&,char)`, így gyengítik az osztályon mért kohéziót és növelik az osztályok közti csatolást. Nem egyértelmű a definíciókból, hogy ezeket a névtér függvényeket figyelembe veszi-e a szerző a metrikák számításakor.

A 3. és 4. fejezetben definiált metrikák nyomán a dolgozat az 5. fejezetben újabb (paraméteres) fogalmi metrikákat vezet be, melyek hibaelőjelző képességét empirikusan igazolja a szerző.

A dolgozat második témaköre maga is két fejezetből áll. A 6. fejezet egy egyesített publikus Java hiba adatbázis létrehozását írja le. A mérnöki-technikai megoldáson túl a munka tudományos értékét az adja, hogy a szerző kiegészítette az adatokat egységes metrika értékekkel és megmutatta, hogy a létrejött egyesített adatbázis jobban használható hiba előrejelző gépi tanuló modellek készítéséhez, mint a korábbi különálló adatbázisok. Ugyanakkor erősen szélsőséges eredmények is előfordultak.

A 7. fejezet a ma oly népszerű mély neurális hálók hibadetektálási lehetőségeinek lehetőségeit tárgyalja. A módszer osztály-finomságú hiba előrejelzést alkalmaz. A fejezet részletes módszertant ad a mély neurális hálók hiba előrejelzésben történő alkalmazására és hiperparaméterek optimális értékeinek megállapítására.

A tézisek értékelése

Az összes tézist elfogadom tudományos eredménynek. A hatodik fejezetben szereplő T4 tézisben az egyesített hibaadatbázist mérnöki-technikai eredménynek tekintem, a kidolgozott egységes metrikákat és az egységes adatbázis eredményességének igazolását tudományos eredménynek fogadom el.

Kérdések

1. A fogalmi metrika kiszámításakor figyelembe veszi-e az adott osztály interfészéhez tartozó, nem tagfüggvényeket?

2. Mennyire befolyásolja a fogalmi metrikát, ha a fejlesztő csapat anyanyelve nem az angol, vagy ha a fejlesztést időközben egy másik team veszi át? Lehet-e a valamilyen módon ellenőrizni a kommentek vagy névválasztás minőségét?

3. Ugyanolyan súllyal lehet-e figyelembe venni az egyes gyakori, általánosan használt változó (i,n) vagy metódusneveket (empty, size), mint a domén-specifikusakat? Érdemes-e bevezetni valami treshold értéket a nevek minimális hosszára?

4. Felmerült-e, hogy pl. a függvényparaméterek véletlen felcserélését detektáló eszközökhöz hasonlóan szinonima-szótárt használjanak a hasonló jelentésű/használatú nevek azonosítására? Pl. a size és length nevek azonos fogalmakat takarhatnak.

5. Felmerült-e, hogy a szoftver forráskódján kívüli információkat elemezzenek a csatolási (coupling) mérőszámok vizsgálatánál? Ilyen lehet pl. a verziókezelő rendszerekből vagy a dokumentációból kinyert információ.

6. Az összes hibadetekciós módszer eredményezhet fals pozitív jelentéseket. Emiatt egyelőre egy szakértőnek kell elbírálnia ezeket a potenciális találatokat. Ez többnyire lassú, költséges, nemegyszer fájdalmas folyamat. Mennyire explicitek a 7. fejezetben leírt hiba predikciók a hiba fajtája illetve helye kapcsán?

Összegzés

A disszertációban szereplő tételek összességükben alkalmasak az „MTA doktora” cím elnyerésére. Így a fentiek alapján javaslom a nyilvános vita lefolytatását.

Budapest, 2024. május 17.

Porkoláb Zoltán

Porkoláb Zoltán