Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Balogh Gergő
Szoftverfejlesztés Tanszék
Szegedi Tudományegyetem

Szeged, 2020

Témavezető:
Dr. Beszédes Árpád

A Ph.D. értekezés téziseinek összefoglalása

Szegedi Tudományegyetem
Informatikai Doktori Iskola

Bevezetés

A szoftverfejlesztés során számtalan elvont fogalmat használunk. Ezek közül néhányat a felhasználók is jól ismernek. Általában azokat, melyek szoros kapcsolatban állnak a vég-felhasználók számára is elérhető funkciókkal. Ezek az elvont elemek leírhatják magát a terméket vagy a fejlesztés folyamatát. Például a felhasználó képes megbecsülni az elérhető funkciók számát és az ezek kifejlesztéséhez szükséges időt (mely végül a szoftver költségében nyilvánul meg). De különböző felhasználói felületek általában csak egy kis részét jelentik annak a hatalmas elvont elem és tulajdonság halmaznak, melyet szoftver rendszernek nevezünk.

Bár, az összes szakember missziója azonos: létrehozni vagy tovább fejleszteni a meglévő szoftvert; mégis az egyéni céljaik eltérhetnek egymáséitól. Például a menedzser arra törekszik hogy minél több új funkció készüljön el a lehető legrövidebb idő alatt. Ezzel szemben a fejlesztők szeretnének különböző technikai és esztétikai szabályokat betartani, melyek általában növelik a fejlesztési időt, de később csökkentik a karbantartási költségeket.

A kutatók felelőssége folyamatosan nő, hiszen az ő feladatuk azoknak a módszereknek és technológiáknak a kifejlesztése, melyet a különböző szakemberek használnak a munkájuk során. Ezeknek a módszereknek (és a hozzájuk kapcsolódó eszközöknek) figyelembe kell venniük a szoftver rendszerek összetettségét és a fejlesztésük egyre gyorsuló ütemét. A gyakorlatban is használható navigációs lehetőségekre van szükség, melyek képesek rámutatni a rendszer olyan pontjaira, amik később hibás működést eredményezhetnek. Az ilyen részek kézi vizsgálatának csökkentésével a különböző fél- és teljesen automatizált javító és megelőző technikák (mint például az újratervezés) feltehetőleg képesek csökkenteni a fejlesztése szánt időt.

Kihívások

Kutatóként a felelősségem, hogy segítsem a különböző szakembereket a közös missziójuk során, úgy, hogy közben ne akadályozzam őket az egyéni céljaik elérésében. Ezt a különböző eszközök és folyamatok tökéletesítése, valamint a kezdők támogatása során érhetem el. Ezeket az elveket figyelembe véve, a következő listában közzé tett általános kihívások megoldását tűztem ki célul a kutatásom során.

1. kihívás: Szoftverek megértése. *A kezdő és új csapattagoknak át kell látniuk a korábban készített, nagy-méretű kódbázist, valamint meg kell ismerkedniük a fejlesztés során használt elvont fogalmakkal. Továbbá szükséges egy*

olyan módszer, mellyel a tapasztalt fejlesztők hatékonyan tudnak navigálni a komplex forráskód szerkezetekben.

2. kihívás: Hibakeresés. *A fejlesztőknek és tesztelőknek meg kell találniuk azokat a részeket a szoftverben és hozzá kapcsolódó teszt csomagban, amik hibás működést okozhatnak, vagyis azokat, melyek sértik a szakmai tapasztalatok által megalapozott irányelveket.*

3. kihívás: Költségbecslés. *A fejlesztési folyamatok javítása érdekében a menedzsernek folyamatosan figyelemmel kell kísérnie ezek tulajdonságait. Azonban a korábban elterjedt mennyiségi mérőszámok általában nem megfelelőek a kreatív munkát igénylő folyamatok jellemzésére, mint amilyen a szoftverfejlesztés.*

4. kihívás: Program szerkezetének elemzése. *A szoftverfejlesztéssel kapcsolatos kutatások gyakran támaszkodnak a különböző elemek hálózatának analizálására. Például, a szoftver elemzés gyakran használja a teszt és a forráskód közötti kapcsolatokat, azonban ezek nem mindig kerülnek közvetlenül jelölésre.*

Tézispontok

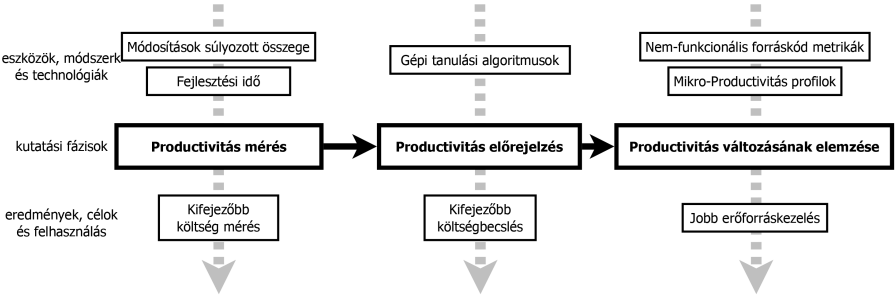
A szerző tudományos eredményei három fő tézispontba csoportosíthatóak. Az ezekbe tartozó altézispontok és a szerző tudományos publikációi közötti kapcsolatot az 1. ábra mutatja.

	1. tézispont csoport	**2. tézispont csoport**	**3. tézispont csoport**
[8]	1.1, 1.2		
[10]	1.3		
[3]	1.4		
[6]		2.1	
[1]		2.1, 2.3	
[5]		2.2	
[7]		2.3	
[2]		2.4	
[9]			3.1, 3.2
[11]			3.2, 3.3
[12]			3.2, 3.3
[4][1]			3.3

1. táblázat. Tézispontok és publikációk kapcsolata

[1] Ez a cikk jelenleg bírálat alatt van.

T1: A szoftverfejlesztői csapatok produktivitásának mérése és előrejelzése. A kutatásom célja, hogy válaszokat adjak a projekt menedzsmentet érintő két fő kihívás – a költségbecslés és az elpazarolt erőforrások kezelése – által támasztott kihívásokra (1. ábra).



1. ábra. A produktivitással kapcsolatos kutatásaink fázisai

A több eltérő produktivitás definíció közül, felhasználtuk azt mely a megtermelt haszon és a befektetett erőforrás arányaként írja le a fogalmat. Pontosabban, a módosítások súlyozott számának (TMod) és a befektetett időnek (DT) a hányadosaként fejeztük ki a produktivitás mértékét (MEff).

$$\text{produktivitás} = \frac{\text{kimenet}}{\text{bemenet}} = \frac{\text{TMod}}{\text{DT}} = \text{MEff} \tag{1}$$

T1.1: ***A produktivitás metrika ami figyelembe veszi a módosítások típusát nagyobb kifejezőerővel rendelkezik.*** A kutatásaim [8] során két új metrikát definiáltam, melyek a módosítások súlyozott száma (TMod) és az azt felhasználó egységnyi erőforrással végzett módosítás mennyisége (MEff). Ezek a finomhangolható mérőszámok nagyobb kifejezőerővel rendelkeznek, mint a produktivitás mérése során korábban használt módosított sorok száma alapú változatok. A fejlesztők általában logikai egységekre bontják mind a programot, mind az azon elvégzendő feladatokat, és csak nagyon ritkán végeznek sorszintű elemzést a munkájuk közben. Az általunk bevezetett új metrikák, jobban tükrözik ezt a gondolkodásmódot.

A MEff metrika, kifejezi az egységnyi idő által elvégzett módosítások mértékét. A fejlesztői produktivitás mértékének illusztrálásához az 1. kódrészletet fogjuk használni. Ennek a módosított változata (2. kódrészlet) két változást tartalmaz, mely három független sorban található. Az első egy „visszaté-

résí érték változás" a 2. sorban, a második egy a „metódus megvalósítást érintő változás" a 4. és a 6. sorban. A példa kedvéért, tegyük fel hogy e változtatások alkalmazása 8 percet vett igényebe. Ezen adatok alapján meghatározható a MEff értéke.

$$\frac{1 \text{ visszatérési érték változás} + 1 \text{ metódus megvalósítás változás}}{8 \text{ perc}} = 0.25$$

Vegyük észre, hogy ez az érték eltér az egyszerű sorok számán alapuló méréstől, mely 3 változott sor/8 perc = 0.375. Továbbá különböző súlyok választása segítségével kifejezhetjük az egyes módosítás típusok elvégzéséhez szükséges erőfeszítés mértékét.

1. kódrészlet. Eredeti verzió

```
1  class IntSet {
2    protected double Find(double limit) {
3      for (int i=0; i<Count(); i++) {
4        double current=Items[i];
5        if (current>limit) {
6          return current;
7        }
8      }
9    }
10 }
```

2. kódrészlet. Módosított verzió

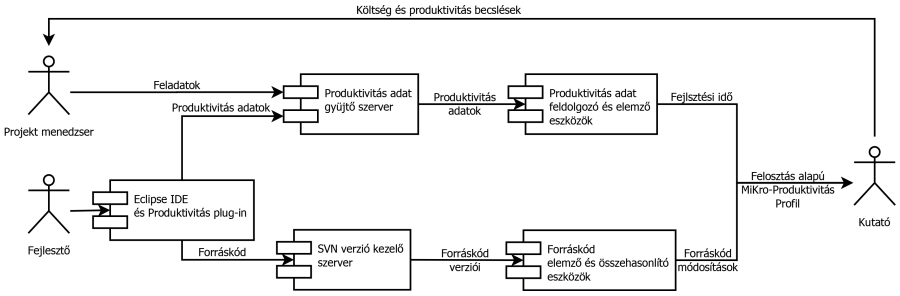
```
1  class IntSet {
2    protected int ✍ Find(double limit) {
3      for (int i=0; i<Count(); i++) {
4        int ✍ current = Items[i];
5        if (current>limit)     {
6          return i; ✍
7        }
8      }
9    }
10 }
```

T1.2: ***A produktivitást előrejelző modell hatékonysága növelhető a módosítás típusainak figyelembevételével.*** Sikeresen növeltem a hatékonyságát a korábbi folyamat és termék metrikákon alapuló költség-előrejelző modellnek az előző tézispontban ismertetett módszerek bevezetésével. Kutatásaim [8] kimutatták, hogy az előrejelzés hatékonysága jelentősen, átlagosan 50%-ról 70%, nőtt az F-mérték szerint.

Az előrejelző modell a döntési fa típusú gépi tanulási algoritmuson alapszik. Továbbá felhasználtunk evolúciós algoritmusokat a modell finomhangolása során. Ebben a lépésben az egyes egyedek életképessége a különböző súlyokkal vett előrejelző modell hatékonysága.

Az adatokat megközelítőleg 800 verzióból gyűjtöttük, melyek egy 75 napos fejlesztési periódust fedtek le. Vizsgáltunk ipari és K+F projekteket is. Ezek többsége Java nyelven íródott, Java EE 6 és Seam 2 technológiák felhasználásával.

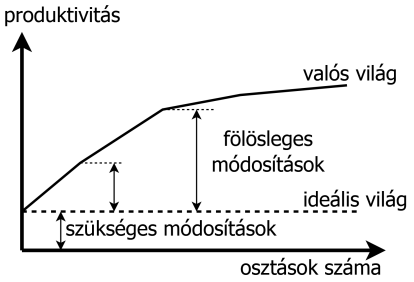
T1.3: ***Az elpazarolt erőforrások mértékének mérése segítséget nyújt a menedzsereknek, hogy javítsák a fejlesztési folyamataikat.*** A produktivitás időbeli változásait egy közepes méretű J2EE projekt vizsgálata során tanulmányoztam [10], hét hónapot átfogó időszakban, 17 fejlesztő bevonásával. A kísérlet során gyűjtött adatok alapján a szakemberek képesek voltak a fejlesztési folyamataik problémás pontjainak lokalizálására.



2. ábra. Mérési architektúra

Jelen fázis központi fogalma a Mikro-Produktivitás Profil (MPPD) volt, mely képes jellemezni a produktivitás mértékének változását különböző mérési felbontások szerint (3. ábra). Az MPPD görbe a fejlesztés során felhasznált fölösleges erőforrások mértékét jelzi. Egy ideális világban ennek az értéke zéró, és a görbe egy vízszintes egyenes vonal lenne. A valóságban azonban befolyásolja a pontatlan specifikáció és a fejlesztés során változó követelmények. A görbe meredeksége a fejlesztők által újra-módosított kódrészletek mennyiségét jelzi.

T1.4: ***A fejlesztők és a diákok forráskódjának átlagos minősége (az egyik produktivitást befolyásoló tényező) nem mutat jelentős eltérést az órai feladatok megoldása alapján.*** A kísérlet [3] során dákok és fejlesztők által készített Java nyelvű feladatmegoldások minőségét hasonlítottam



3. ábra. Az MPPD értelmezése

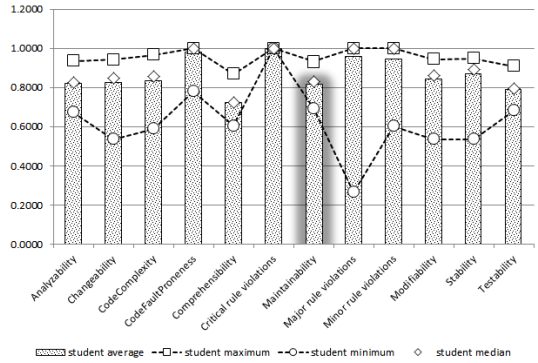
össze nem-funkcionális tulajdonságok segítségével. Az adatok arra engednek következtetni, hogy nincs jelentős eltérés a két csoport munkájának átlagos minősége között, viszont a diákok esetében több kiugró érték is megfigyelhető, amely nagyobb átlag körüli szórást eredményez (4. ábra)[2].

T2: Izgalmas és magával ragadó szoftver és teszt vizualizációs technikák biztosítása. Ez a tézis pont a szoftver rendszerek és a hozzájuk kötődő elemek vizualizálásával kapcsolatos kutatásokat részletezi.

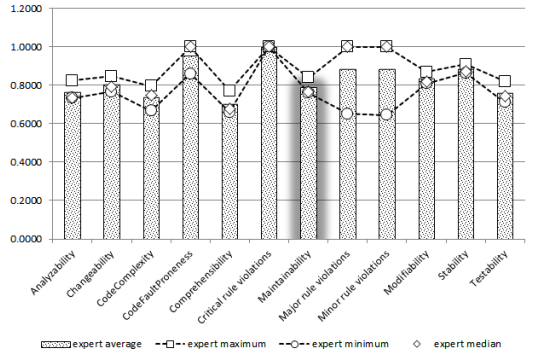
T2.1: ***Nyílt-terű játékok alkalmazása segíti a virtuális városként reprezentált szoftver megértését.*** A fő eredményem az adatok vizuális megjelenítésének összekapcsolása magas grafikai lehetőségeket biztosító játékkal. Az általam bővített vizualizációs megoldás és a hozzá kapcsolódó eszközkészlet, segíti a összetett szoftver rendszerek megértését a fejlesztők és hallgatók számára. Ezt virtuális városok generálásával értem el, mely olyan különböző elvont fogalmakat jelképeznek, mint például a forráskód metrikák.

A vizualizációs folyamat során két szintet használunk az adatok és grafikus elemek kezelésére. Az jelképezendő adatok szintjén, minden entitás rendelkezik egy tulajdonsághalmazzal, melyek például az egyes forráskód elemek metrikáit tartalmazzák. Ezeket az információkat a metafora szintjéhez tartozó grafikus elemek jelenítik meg. A várost alkotó épületek különböző képi megjelenítést befolyásoló tulajdonságokkal rendelkeznek. Ezekhez a tulajdonságokhoz rendeljük az adatok különböző elvont értékeit a vizualizálás során.

[2]A félreértések elkerülése végett a nem-funkcionális metrikák megnevezése során az eredeti angol szakkifejezéseket használom.



(a) A diákok forráskód minőség metrikái



(b) A fejlesztők forráskód minőség metrikái

4. ábra. Magas szintű forráskód metrikák

T2.2: ***A valósághűség mértéke kifejezhető a generált városokat használó szoftver vizualizáció során.*** A kutatásom során bevezetésre került három alacsony- és egy magas-szintű metrika, melyek képesek kifejezni a generált városok különböző jellemzőit és becslést adni a város valósághűségének mértékére. Mind a négy metrika egy felhasználói kérdőív segítségével került kiértékelésre [5]. Az eredmények azt mutatják, hogy lehetséges olyan automatikus módszert definiálni, ami képes megbecsülni a generált és valós városok közti hasonlóságot.

Az alacsony-szintű metrikák a következők: a kompaktság, ami a város

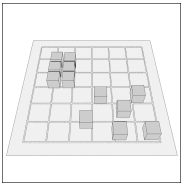
(a) Körletek az épületek csoportosítására

(b) Kertek különböző mennyiségű virágokkal

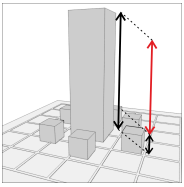
(c) Épületek kertekkel körülvéve

(d) Többféle anyagból épült szintek

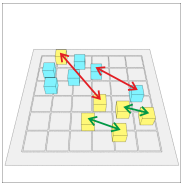
5. ábra. A metafora szint elemei



(a) Kompaktság



(b) Homogenitás



(c) Összekapcsoltság

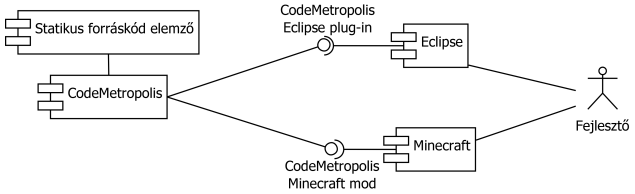
6. ábra. Alacsony-szintű metrikák

térbeli kiterjedését méri; az összekapcsoltság, ami az épületek közötti kis-léptékű koherenciát fejezi ki; és a homogenitás, ami a látkép folytonosságát jelzi.

A kiértékelés során 51 teljes és 20 részlegesen kitöltött kérdőív adatait használtam fel a magas-szintű metrika létrehozása során. A felhasználóknak rangsorolnia kellett a különböző városokat a valósághűségük alapján, vala-

mint el kellett dönteniük, hogy a kapott példák közül melyik mutatja jobban az adott alacsony-szintű tulajdonság értékét. A felhasználók válaszait reprezentáló rangsor kiválasztása során Kendall-tau korrelációs együtthatót és közösség detektáló algoritmust használtam. Az így kapott egyenlőtlenségrendszer relaxált megoldása segítségével határoztam meg a magas-szintű metrika konstruálásához szükséges súlyokat.

T2.3: ***A fejlesztő környezet és a szoftvervizualizáció integrálása segíti a fejlesztőket a program rendszerek megértésében.*** Ebben a fázisban egy új módszert mutattunk be, mely lehetővé tette a közismert Java nyelvű fejlesztéseket támogató rendszer, az Eclipse, és a korábban részletezett városmetaforát használó szoftvervizualizációs program csomag, a CodeMetropolis együttes használatát. A korábbi független használati esetekkel ellentétben, jelen verzió biztosítja, hogy a felhasználó navigálhasson a forráskód és az azt jelképező város elemei között. Ezeket a funkciókat egy Eclipse plug-in és egy Minecraft mod biztosítja.

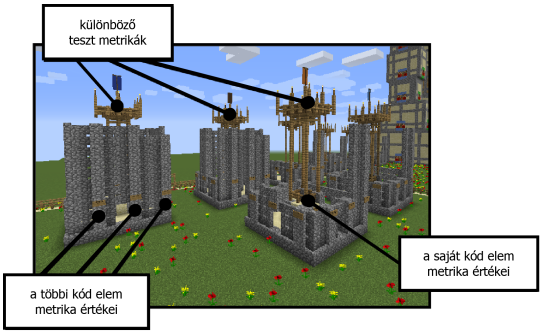


7. ábra. Eclipse és CodeMetropolis integráció

T2.4: ***A város metafora lehetővé teszi teszt metrikák és teszt-kód kapcsolatok vizualizálását.*** Az általunk bevezetett új módszer lehetővé tette hogy kiterjesszük a korábban felhasznált metaforát tesztekkel kapcsolatos adatokkal [2]. Ennek következtében a CodeMetropolis programcsomag keretében egyesítsük a korábbi program struktúrát bemutató és az új teszt minőséget metrikákkal jelző technológiákat.

A módszer alkalmazása során, például a forráskód elemeket jelképező házakat a tesztek tulajdonságait szimbolizáló őrtornyok védelmezik. A tornyok fizikai tulajdonságai, mint a magassága vagy az építési anyaga, teszt-kód csoportok különböző mérőszámait jelzik, például, hogy az adott teszt mekkora mértékben fedte le az vizsgált forráskód elemet.

A megjelenített teszt és forráskód elemek közti kapcsolatokat a korábban említett őrtornyok elhelyezkedése jelzi. A teszt metrikákat jelképező épületek



8. ábra. Az őrtornyok részei

ugyan abba a kertbe kerülnek, mint a vizsgált forráskód elemet reprezentáló házak. A központi torony és a körülötte lévő kerítés magassága különböző metrikák értékeit mutatják az adott teszt és a hozzá kapcsolódó funkció szerint. Az egyes funkciók elkülönítését a különböző építési anyagok segítik. Továbbá a tornyok megfelelő pontjain tájékozódást könnyítő táblákat helyeztünk el.

T3: Figyelmet érdemlő helyek azonosítása a csomaghierarchiában lefedettségi adatok alapján. Ebben a tézispontban összefoglalom a teszt lefedettség analízis során és az ennek felhasználásával elért eredményeket az (egység) tesztek és a hozzá kapcsolódó kód elemek javítása közben.

T3.1: ***A közösség detektáló algoritmusok képesek a teszt és forráskód elemek együttes klaszterezésére.*** A különböző teszt és forráskód analízisek automatizálásának érdekében, egy olyan módszert hoztam létre mely képes tesztek és az általuk ellenőrzött forráskód elemek együttes csoportosítására. Ez a módszer további vizsgálatokat tett lehetővé.

A kutatás során két különböző klaszterező algoritmust definiáltunk a teszt és kód elemek halmazán. Az első a teszt-kód lefedettségi adatokon alapszik és kifejezi a teszt csomag dinamikus, vagyis futás közbeni viselkedését. Az így kapott csoportok összehasonlításra kerültek a második algoritmus által meghatározottakkal, mely a forráskód csomaghierarchiáján alapszik és a statikus teszt-kód kapcsolatokat fejezi ki. A dinamikus csoportosítás meghatározásához közösség detektáló algoritmusokat használtunk a futás közben gyűjtött részletes lefedettségi adatokon. Ez a módszer leegyszerűsítve a a teszteset-metódus lefedettségi mátrixok olyan részeit rendeli egy csoportba, melynek elemei között jelentősen több kapcsolat található mint a csoporton kívül.

T3.2: ***A teszt és kód elemek struktúrájának eltérésinek osztályozása segítséget nyújt a fejlesztők és tesztelők számára a teszt és forráskód minőségének javításában a teszt-kód kapcsolatok helyreállításában.*** Ez a munka az egység tesztek minőségének vizsgálatát teszi lehetővé egy újszerű nézőpontból. A módszer lényege a statikus, csomaghierarchia alapú és a dinamikus, futás közbeni elemzésből származó teszt és kód elem csoportok összehasonlítása.

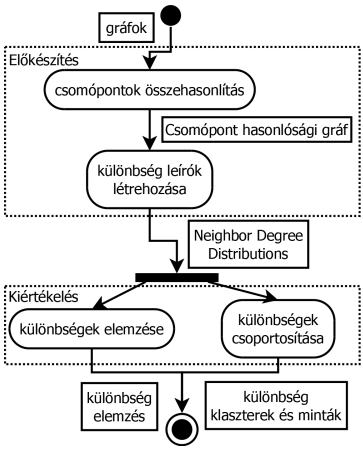
A problémás helyek újratervezésére tett javaslatok megfogalmazása nagy nyílt forráskódú rendszerek elemzésén alapszik, melyek számottevő méretű teszt csomaggal rendelkeznek. A vizsgált rendszerek közepes és nagy Java nyelven íródott programok voltak, melyek tesztelését JUnit keretrendszer segítségével valósították meg. A választás során figyelembe vettük, hogy a rendszerek a méretükhöz képest megfelelő számú teszt esettel rendelkezzenek. Az ezekből gyűjtött adatokat a tesz és kód közötti kapcsolatok helyreállítása során, az adott elemek környezetének leírására használtuk fel. A módszer a statikus és dinamikus elemzések során meghatározott kapcsolatok összehasonlításán alapszik. A hozzárendelések helyreállítására vonatkozó végső döntést a fejlesztő hozza meg az eltéréseket mutató pontok vizsgálata után.

Ezek az eltérések egyfajta gyanús jelekként[3] is értelmezhetőek, melyek olyan pontokat jeleznek ahol a teszek vagy a hozzájuk kapcsolódó forráskód elemek struktúrájában esetleg javításra van szükség. A korrekció szükségességét egyéni mérlegelés után a fejlesztő dönti el. Kutatásunk első fázisa során a saját szakmai tapasztalataink alapján definiáltunk alacsony felbontású eltéréseket leíró mintázatokat. Ezeket a második fázisban nagyobb felbontással rendelkező mintákra bontottuk szét, hogy segítsük a fejlesztőt a hasonló javításokat igénylő helyek felismerésében. A vizsgáltatok a későbbi altézispontban bemutatásra kerülő UniGDA módszertan elemeit használja fel, például a minták definiálására szolgáló dNDD leírókat.

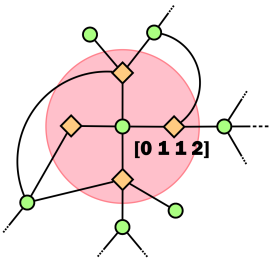
T3.3: ***Általános gráf összehasonlító módszertan biztosítása.*** Az általam bevezetett és definiált általános gráf összehasonlító módszertan az UniGDA nevet viseli. Ez a korábban bemutatott szakterület specifikus különbség elemző technikák kiterjesztése terület független mintákkal és hasonlósági függvényekkel. A kiterjesztés lehetővé teszi tetszőleges gráfok összehasonlítását.

Az UniGDA két fő fázisból áll (9a. ábra). Az első fázis során összehasonlítjuk az elemzendő gráfok minden egyes csomópontját, majd a kapott adatokat egy általános, különbségeket és hasonlóságokat leíró adatszerkezetben tároljuk. Ezt a struktúrát fogjuk a későbbiekben felhasználni az eltérések

[3] Angolul: „bad smell”



(a) A folyamat áttekintése



(b) Példa gráf az DNDD számításához

9. ábra. Általános gráfok közti különbség elemző módszertan

elemzésére és osztályozására a kiértékelési fázisban.

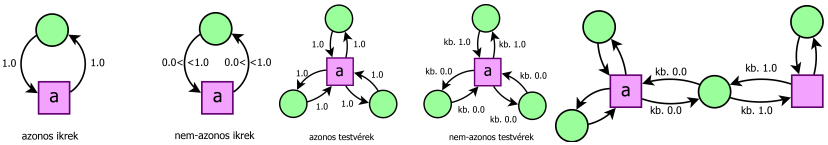
Definiáltunk egy gráf alapú reprezentációt (csomópont hasonlóság gráf, NSG), ami képes minden hasonlósági adat tárolására az eltérések lokalizációjához. Ezen különbségek további elemzése megköveteli hogy képesek legyünk jellemezni őket, vagyis szükségünk van egy eszközre, mely választ tud adni a következő informális kérdésre: *A hasonló elemek hogyan viszonyulnak a többi elemhez?* A probléma megoldása érdekében bevezetésre került egy általános leíró vektor és függvény (DNDD és CNDD), melyek képesek az NSG csomópontjainak szomszédait és kapcsolatukat jellemezni. Ezek a szomszédos csomópontok fokszám szerinti eloszlását írják le. Az NSG esetében, segítségükkel képesek vagyunk jellemezni az egyes csomópontok lokális hasonlóságát más elemekhez képest.

Például a 9b. ábra középső csomópontjának DNDD vektora a következő: $d = (0, 1, 1, 2, 0, 0, \dots)$. A vörös kör az DNDD, hatósugarát jelzi mely, csak a második szomszédságig képes jellemezni a csomópontok környezetét. A vektor szerint a vizsgált csomópont az alábbi szomszédokkal rendelkezik.

$d_1 = 0$ Nincs olyan szomszédja, melynek csak egy kapcsolata van.
$d_2 = 1$ Egy olyan szomszédja van, melynek kettő kapcsolata van, a baloldali.
$d_3 = 1$ Egy szomszédjának van három kapcsolata, a jobb oldalinak.
$d_4 = 2$ Két szomszédos csomópontnak van négy kapcsolata, a felsőnek és az

alsónak.

Az NDD vektorok felhasználása lehetővé tette az elemek hasonlóságának mélyebb elemzését. A leírókat felhasználhatjuk az ún. *hasonlósági minták* definiálása során. Ezek olyan részgráfjai az NSG-nek, melyek képesek leírni a vizsgált csomópont és hozzá hasonló elemek közti kapcsolatok szerkezetét. A 10. ábra több példát is mutat ez ilyen területfüggetlen mintákra. Nehéz általános, szakterület független jelentéstartalommal felruházni az egyes összetettebb mintákat. De például a mostoha minta utalhat az elemzés során rosszul megválasztott hasonlósági függvény használatára. Például, ha forráskódban szereplő metódusokat minden lehetséges tulajdonságuk szerint hasonlítjuk össze, beleértve a láthatóságukat is (private, public, protected), akkor elkerülhetetlen hogy több olyan elemet is kapunk, melyek csak nagyon kis mértékben lesznek hasonlóak a vizsgálthoz.



(a) Ikrek (b) Testvérek (c) Mostoha

10. ábra. Hasonlósági minták

Az UniGDA alkalmazása során mind a megfelelő hasonlósági függvény kiválasztása és a kapott minták értelmezése is a kihívások közé tartozik. Ezek mélyebb ismereteket igényelnek az adott szakterülettel kapcsolatban. A tapasztalataim arra engednek következtetni, hogy ezen komponensek meghatározhatók manuális kiértékelés és elemzés segítségével. Köztudott, hogy a szakértői tudás összegyűjtése és kiértékelése hosszadalmas és költséges feladat. Azonban az általam bemutatott UniGDA módszertan lehetővé teszi, hogy ezeket elég legyen egyszer elvégezni, hogy a további kutatások során többször felhasználhassuk őket.

Bibliográfia

[1] Gergo Balogh és Arpad Beszedes. „CodeMetrpolis—A minecraft based collaboration tool for developers”. *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*. IEEE. 2013, 1–4. old.

[2] Gergo Balogh és tsai. „Using the City Metaphor for Visualizing Test-Related Metrics". *1st International Workshop on Validating Software Tests*. 2016.

[3] Gergő Balogh. „Comparison of Software Quality in the Work of Children and Professional Developers Based on Their Classroom Exercises". *International Conference on Computational Science and Its Applications*. Springer, Cham. 2015, 36–46. old.

[4] Gergő Balogh. „First Steps towards a Methodology for Unified Graph's Discrepancy Analysis". submittted for review to 13th International Conference of Graph Transformation, (part of STAF 2020).

[5] Gergő Balogh. „Validation of the city metaphor in software visualization". *International Conference on Computational Science and Its Applications*. Springer, Cham. 2015, 73–85. old.

[6] Gergő Balogh és Arpad Beszedes. „CodeMetropolis-code visualisation in MineCraft". *Source Code Analysis and Manipulation (SCAM), 2013 IEEE 13th International Working Conference on*. IEEE. 2013, 136–141. old.

[7] Gergő Balogh, Attila Szabolics és Arpád Beszédes. „CodeMetropolis: Eclipse over the city of source code". *Source Code Analysis and Manipulation (SCAM), 2015 IEEE 15th International Working Conference on*. IEEE. 2015, 271–276. old.

[8] Gergő Balogh, Ádám Zoltán Végh és Árpád Beszédes. „Prediction of Software Development Modification Effort Enhanced by a Genetic Algorithm". *SSBSE Fast Abstract track* (2012), 1–6. old.

[9] Gergő Balogh és tsai. „Are My Unit Tests in the Right Package?": *Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on*. IEEE. 2016, 137–146. old.

[10] Gergő Balogh és tsai. „Identifying wasted effort in the field via developer interaction data". *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE. 2015, 391–400. old.

[11] Tamás Gergely és tsai. „Analysis of Static and Dynamic Test-to-code Traceability Information". *Acta Cybernetica* 23.3 (2018), 903–919. old.

[12] Tamás Gergely és tsai. „Differences between a static and a dynamic test-to-code traceability recovery method". *Software Quality Journal* (2018), 1–26. old.

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Identifying wasted effort in the field via developer interaction data" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Micro-Productivity Profilok időbeli felbontásának meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A korábbi kutatások során publikált Micro-Productivity Profilok értelmezése és alkalmazása a súlyozott módosítás alapú produktivitás mérés során
- A fejlesztő csapat produktivitásának időbeli változásainak vizsgálata frekvencia térben

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Esemény naplózó rendszer megvalósítása
- Vizsgált rendszerek alapadatainak mérése és osztályozása
- Detektált fejlesztési fázisok és ütemezése értelmezése

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Identifying wasted effort in the field via developer interaction data". *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on.* IEEE. 2015, 391–400. old.

2020. március 31.

Antal Gábor

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőségőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Analysis of Static and Dynamic Test-to-code Traceability Information" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Statikus és dinamikus teszt-kód csoportosítás közti eltérések értelmezése környezeti információként a hozzárendelés kapcsolatok helyreállítása során

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Analysis of Static and Dynamic Test-to-code Traceability Information". *Acta Cybernetica* 23.3 (2018), 903–919. old.

2020. április 1.

.....
Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Are My Unit Tests in the Right Package?" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Klaszterhasonlósági mintázatok pontosítása és finomhangolása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Teszt és forráskód elemek egyidejű klaszterezésének kifejlesztése
- Statikus és dinamikus teszt és forráskód elem klasztereket összehasonlító módszer kidolgozása
- Kezdeti klaszterhasonlósági mintázatok definiálása
- Klaszterhasonlósági mintázatok automatikus detektálása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Újratervezési minták pontosítása és finomhangolása
- Használati esetek és motivációs forgatókönyvek definiálása

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Are My Unit Tests in the Right Package?": *Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on.* IEEE. 2016, 137–146. old.

2020. április 1.

.....
Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „CodeMetropolis-code visualisation in MineCraft" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket. Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Gyakorlati használati esetek meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A város metafora továbbfejlesztett változatának alkalmazása szoftvervizualizáció során
- Korszerű grafikus motor használata a részletgazdag generált város megjelenítésére
- Szoftvervizualizáció megvalósítása interaktivitást biztosító játék-motor segítségével

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és Arpad Beszedes. „CodeMetropolis-code visualisation in MineCraft". *Source Code Analysis and Manipulation (SCAM), 2013 IEEE 13th International Working Conference on.* IEEE. 2013, 136–141. old.

2020. április 1.

Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „CodeMetropolis: Eclipse over the city of source code" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket. Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A város alapú szoftvervizualizáció és a fejlesztő környezetek közti kapcsolódási lehetőségek felderítése
- Alapvető interaktív navigációt és megértést könnyítő funkciók definiálása

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh, Attila Szabolics és Arpád Beszédes. „CodeMetropolis: Eclipse over the city of source code". *Source Code Analysis and Manipulation (SCAM), 2015 IEEE 15th International Working Conference on.* IEEE. 2015, 271–276. old.

2020. április 1.

.....
Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „CodeMetrpolis—A minecraft based collaboration tool for developers" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A város metafora alapú szoftvervizualizáció együttműködés célú felhasználásának feltérképezése
- Az említett megvalósítás helyének meghatározása a fejlesztési munka menetben

Az érintett cikk teljes hivatkozása a következő:

> Gergo Balogh és Arpad Beszedes. „CodeMetrpolis—A minecraft based collaboration tool for developers". *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on.* IEEE. 2013, 1–4. old.

2020. április 1.

.....
Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Differences between a static and a dynamic test-to-code traceability recovery method" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása
- A definiált minták detektálását végző módszerek kifejlesztése és megvalósítása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Az azonosított hasonlósági (al-)minták finomítása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Diszkrét csomópontok szomszédjainak fokszám szerinti eloszlását leíró vektorok definíciójának bevezetése
- A korábbi statikus és dinamikus tesz-kód csoportok eltéréseiben azonosított mintázatok további al-mintákra bontása
- A mérési folyamatok végrehajtása és felügyelete

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Differences between a static and a dynamic test-to-code traceability recovery method". *Software Quality Journal* (2018), 1–26. old.

2020. április 1.

.....
Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Identifying wasted effort in the field via developer interaction data" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Micro-Productivity Profilok időbeli felbontásának meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A korábbi kutatások során publikált Micro-Productivity Profilok értelmezése és alkalmazása a súlyozott módosítás alapú produktivitás mérés során
- A fejlesztő csapat produktivitásának időbeli változásainak vizsgálata frekvencia térben

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Identifying wasted effort in the field via developer interaction data". *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on.* IEEE. 2015, 391–400. old.

2020. április 1.

Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Prediction of Software Development Modification Effort Enhanced by a Genetic Algorithm" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Előrejelző algoritmus paramétereinek becslése és meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Súlyozott módosítás alapú produktivitás mérés módszerének kidolgozása
- Fejlesztői teljesítményt mérő metrika definiálása
- Evolúció és genetikus előrejelző algoritmus definiálása

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh, Ádám Zoltán Végh és Árpád Beszédes. „Prediction of Software Development Modification Effort Enhanced by a Genetic Algorithm". *SSBSE Fast Abstract track* (2012), 1–6. old.

2020. április 1.

Beszédes Árpád

[1]Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Témavezetői nyilatkozat

Alulírott, Dr. Beszédes Árpád, mint Balogh Gergő doktorjelölt témavezetője kijelentem, hogy a jelölt „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című értekezésében felhasznált eredmények a jelölt saját hozzájárulását tükrözik.

2020. április 1.

.....
Dr. Beszédes Árpád

[1]Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Using the City Metaphor for Visualizing Test-Related Metrics" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Teszt esetek város metafora alapú megjelenítésének definiálása
- Korábban virtualizált forráskód és teszt esetet jelképező elemek közti grafikus kapcsolat megteremtése

Az érintett cikk teljes hivatkozása a következő:

> Gergo Balogh és tsai. „Using the City Metaphor for Visualizing Test-Related Metrics". *1st International Workshop on Validating Software Tests.* 2016

2020. április 1.

.....
Beszédes Árpád

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Analysis of Static and Dynamic Test-to-code Traceability Information" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Statikus és dinamikus teszt-kód csoportosítás közti eltérések értelmezése környezeti információként a hozzárendelés kapcsolatok helyreállítása során
- Az automatikus mérések alatt detektált eltérések manuális vizsgálata
- A kézi vizsgálat és a felismert hasonlósági minták alapján újratervezési javaslatok megfogalmazása az teszt-kód kapcsolatok helyreállítása érdekében

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Analysis of Static and Dynamic Test-to-code Traceability Information". *Acta Cybernetica* 23.3 (2018), 903–919. old.

2020. április 1.

Gergely Tamás

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Are My Unit Tests in the Right Package?" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Klaszterhasonlósági mintázatok pontosítása és finomhangolása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Teszt és forráskód elemek egyidejű klaszterezésének kifejlesztése
- Statikus és dinamikus teszt és forráskód elem klasztereket összehasonlító módszer kidolgozása
- Kezdeti klaszterhasonlósági mintázatok definiálása
- Klaszterhasonlósági mintázatok automatikus detektálása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Javítási javaslatok és lehetséges újratervezési minták vázlatának meghatározása

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Are My Unit Tests in the Right Package?": *Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on*. IEEE. 2016, 137–146. old.

2020. április 1.

.....
Gergely Tamás

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Using the City Metaphor for Visualizing Test-Related Metrics" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Teszt esetek város metafora alapú megjelenítésének definiálása
- Korábban virtualizált forráskód és teszt esetet jelképező elemek közti grafikus kapcsolat megteremtése

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Teszt esetekhez kapcsolódó metrikák grafikus leképezésének meghatározása
- Megjelenített objektumok (épületek) jelentésének pontosítása és gyakorlati értelmezésének megkönnyítése

Az érintett cikk teljes hivatkozása a következő:

> Gergo Balogh és tsai. „Using the City Metaphor for Visualizing Test-Related Metrics". *1st International Workshop on Validating Software Tests*. 2016

2020. április 1.

Gergely Tamás

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Differences between a static and a dynamic test-to-code traceability recovery method" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Az azonosított hasonlósági (al-)minták finomítása
- A definiált minták detektálását végző módszerek kifejlesztése és megvalósítása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- A korábbi statikus és dinamikus tesz-kód csoportok eltéréseiben azonosított mintázatok további al-mintákra bontása

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Differences between a static and a dynamic test-to-code traceability recovery method". *Software Quality Journal* (2018), 1-26. old.

2020. április 1.

Gergely Tamás

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Are My Unit Tests in the Right Package?" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Klaszterhasonlósági mintázatok pontosítása és finomhangolása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Teszt és forráskód elemek egyidejű klaszterezésének kifejlesztése
- Statikus és dinamikus teszt és forráskód elem klasztereket összehasonlító módszer kidolgozása
- Kezdeti klaszterhasonlósági mintázatok definiálása
- Klaszterhasonlósági mintázatok automatikus detektálása

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Are My Unit Tests in the Right Package?": *Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on.* IEEE. 2016, 137–146. old.

2020. április 1.

Gyimóthy Tibor

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Differences between a static and a dynamic test-to-code traceability recovery method" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

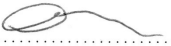
A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Az azonosított hasonlósági (al-)minták finomítása
- A definiált minták detektálását végző módszerek kifejlesztése és megvalósítása

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Differences between a static and a dynamic test-to-code traceability recovery method". *Software Quality Journal* (2018), 1–26. old.

2020. április 1.



Gyimóthy Tibor

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Identifying wasted effort in the field via developer interaction data" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Micro-Productivity Profilok időbeli felbontásának meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

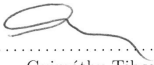
A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A korábbi kutatások során publikált Micro-Productivity Profilok értelmezése és alkalmazása a súlyozott módosítás alapú produktivitás mérés során
- A fejlesztő csapat produktivitásának időbeli változásainak vizsgálata frekvencia térben

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Identifying wasted effort in the field via developer interaction data". *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on.* IEEE. 2015, 391–400. old.

2020. április 1.



Gyimóthy Tibor

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Using the City Metaphor for Visualizing Test-Related Metrics" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

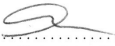
A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Teszt esetek város metafora alapú megjelenítésének definiálása
- Korábban virtualizált forráskód és teszt esetet jelképező elemek közti grafikus kapcsolat megteremtése

Az érintett cikk teljes hivatkozása a következő:

> Gergo Balogh és tsai. „Using the City Metaphor for Visualizing Test-Related Metrics". *1st International Workshop on Validating Software Tests*. 2016

2020. április 1.



Gyimóthy Tibor

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

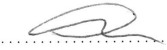
Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Analysis of Static and Dynamic Test-to-code Traceability Information" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Analysis of Static and Dynamic Test-to-code Traceability Information". *Acta Cybernetica* 23.3 (2018), 903–919. old.

2020. április 1.



Gyimóthy Tibor

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Analysis of Static and Dynamic Test-to-code Traceability Information" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Statikus és dinamikus teszt-kód csoportosítás közti eltérések értelmezése környezeti információként a hozzárendelés kapcsolatok helyreállítása során
- Az automatikus mérések alatt detektált eltérések manuális vizsgálata
- A kézi vizsgálat és a felismert hasonlósági minták alapján újratervezési javaslatok megfogalmazása az teszt-kód kapcsolatok helyreállítása érdekében

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Analysis of Static and Dynamic Test-to-code Traceability Information". *Acta Cybernetica* 23.3 (2018), 903–919. old.

2020. március 31.

.....Horváth Ferenc.....
Horváth Ferenc

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőségőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Differences between a static and a dynamic test-to-code traceability recovery method" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása
- A definiált minták detektálását végző módszerek kifejlesztése és megvalósítása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Az azonosított hasonlósági (al-)minták finomítása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- A korábbi statikus és dinamikus tesz-kód csoportok eltéréseiben azonosított mintázatok további al-mintákra bontása
- A mérési folyamatok végrehajtása és felügyelete

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Differences between a static and a dynamic test-to-code traceability recovery method". *Software Quality Journal* (2018), 1–26. old.

2020. március 31.

.....
Horváth Ferenc

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőségőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management" [1] című disszertációját. A disszertációban szereplő és az „CodeMetropolis: Eclipse over the city of source code" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket. Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A város alapú szoftvervizualizáció és a fejlesztő környezetek közti kapcsolódási lehetőségek felderítése
- Alapvető interaktív navigációt és megértést könnyítő funkciók definiálása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Interaktív navigációt és megértést könnyítő funkciók gyakorlati szempont alapján történő finomítása és pontosítása
- Az integrációhoz szükséges szoftverkomponensek tervezése és implementálása

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh, Attila Szabolics és Arpád Beszédes. „CodeMetropolis: Eclipse over the city of source code". *Source Code Analysis and Manipulation (SCAM), 2015 IEEE 15th International Working Conference on.* IEEE. 2015, 271–276. old.

2020. március 31.

Szabolics Attila

Szabolics Attila

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőségőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Analysis of Static and Dynamic Test-to-code Traceability Information" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Statikus és dinamikus teszt-kód csoportosítás közti eltérések értelmezése környezeti információként a hozzárendelés kapcsolatok helyreállítása során
- Az automatikus mérések alatt detektált eltérések manuális vizsgálata
- A kézi vizsgálat és a felismert hasonlósági minták alapján újratervezési javaslatok megfogalmazása az teszt-kód kapcsolatok helyreállítása érdekében

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Analysis of Static and Dynamic Test-to-code Traceability Information". *Acta Cybernetica* 23.3 (2018), 903–919. old.

2020. április 1.

Vancsics Béla

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Differences between a static and a dynamic test-to-code traceability recovery method" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Az azonosított hasonlósági (al-)minták finomítása
- A definiált minták detektálását végző módszerek kifejlesztése és megvalósítása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- A korábbi statikus és dinamikus tesz-kód csoportok eltéréseiben azonosított mintázatok további al-mintákra bontása
- A mérési folyamatok végrehajtása és felügyelete

Az érintett cikk teljes hivatkozása a következő:

> Tamás Gergely és tsai. „Differences between a static and a dynamic test-to-code traceability recovery method". *Software Quality Journal* (2018), 1–26. old.

2020. április 1.

Vancsics Béla

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Identifying wasted effort in the field via developer interaction data" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Micro-Productivity Profilok időbeli felbontásának meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A korábbi kutatások során publikált Micro-Productivity Profilok értelmezése és alkalmazása a súlyozott módosítás alapú produktivitás mérés során
- A fejlesztő csapat produktivitásának időbeli változásainak vizsgálata frekvencia térben

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Fejlesztők aktivitásának begyűjtése
- Esemény naplózó rendszer megtervezése és létrehozásának felügyelete
- Nyers adatok rendszerezése

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Identifying wasted effort in the field via developer interaction data". *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE. 2015, 391–400. old.

2020. március 31.

.....
Végh Ádám Zoltán

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőségbiztosítás menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Prediction of Software Development Modification Effort Enhanced by a Genetic Algorithm" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Előrejelző algoritmus paramétereinek becslése és meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- Súlyozott módosítás alapú produktivitás mérés módszerének kidolgozása
- Fejlesztői teljesítményt mérő metrika definiálása
- Evolúció és genetikus előrejelző algoritmus definiálása

A következő eredményekben az én hozzájárulásom volt a meghatározó:

- Fejlesztők aktivitásának begyűjtése
- Esemény naplózó rendszer megtervezése és létrehozásának felügyelete
- Nyers adatok rendszerezése

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh, Ádám Zoltán Végh és Árpád Beszédes. „Prediction of Software Development Modification Effort Enhanced by a Genetic Algorithm". *SSBSE Fast Abstract track* (2012), 1–6. old.

2020. március 31.

.....
Végh Ádám Zoltán

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőségbiztosítás menedzsment során

Társszerzői nyilatkozat

Kijelentem, hogy ismerem Balogh Gergő PhD fokozatra pályázó „Utilizing static and dynamic software analysis to aid cost estimation, software visualization, and test quality management"[1] című disszertációját. A disszertációban szereplő és az „Identifying wasted effort in the field via developer interaction data" cikkekben publikált közös eredményekre vonatkozóan kijelentem a következőket.
Az alábbi eredményekhez való hozzájárulásunk oszthatatlan:

- Micro-Productivity Profilok időbeli felbontásának meghatározása
- Eredmények kiértékelése
- Kutatás dokumentálása és bemutatása

A következő eredményekben a pályázó hozzájárulása volt a meghatározó:

- A korábbi kutatások során publikált Micro-Productivity Profilok értelmezése és alkalmazása a súlyozott módosítás alapú produktivitás mérés során
- A fejlesztő csapat produktivitásának időbeli változásainak vizsgálata frekvencia térben

Az érintett cikk teljes hivatkozása a következő:

> Gergő Balogh és tsai. „Identifying wasted effort in the field via developer interaction data". *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE. 2015, 391–400. old.

2020. április 1.

.....
Vidács László

[1] Statikus és dinamikus forráskód elemzés felhasználása költségbecslés, szoftver vizualizáció, és teszt minőség menedzsment során