2022

Hierarchical Delta Debugging osztási tényezőjének vizsgálata

BSc Szakdolgozat Téma

Vince Dániel

Szoftverfejlesztés Tanszék | Dugonics Tér 13.

Motiváció

Véletlenszerű tesztesetgenerálás célja egy szoftver biztonságtechnikai tesztelése generált bemenetekkel, hibás viselkedést remélve (pl. memóriaszivárgás). Egy-egy hibát találva, a javítást kivitelező szoftverfejlesztőnek először meg kell találnia a hibát kiváltó tényleges okot.

Ezt a feladatot hivatott megkönnyíteni az automatikus tesztesetredukció diszciplínája, mely algoritmikusan csökkenti a hibát előidéző tesztesetet, annak bizonyos tulajdonságait megtartva. Az első, és mai napig aktívan használt algoritmus a *Delta Debugging* (*DDMin*), mely a bemenetet elemi egységekre bontja, majd ezek különböző konfigurációival próbálja csökkenteni a teszteset méretét. Amennyiben egy bemenet struktúrával rendelkezik (pl. programozási nyelvek (C, C++, C#)), a *DDMin* sok esetben megsérti ezt a struktúrát, így a reprodukálandó hibát nem képes előidézni, mivel a tesztelt szoftver belső szerkezetéig nem jut el a végrehajtás. Ezt a problémakört oldja meg a *Hierarchical Delta Debugging* (*HDD*) algoritmus, mely a tesztesetből egy fát épít, majd a fa szintjein át haladva, annak csomópontjain futtatja a DDMin algoritmust.

Feladat leírása

A *DDMin* algoritmus az elemi egységek konfigurációit kettő hatványával osztja (azaz először 2 részre, majd 4-re, ezután 8-ra stb.), mely nem minden esetben eredményezheti a leghatékonyabb redukciót. Egy közelmúltban publikált tanulmány [1] már vizsgálta az algoritmus viselkedését különböző osztási tényezőkkel, eredményei alapján felhasználás során megéri az algoritmust nem 2 alapú osztási tényezővel használni.

A szakdolgozó feladata implementálni az algoritmusokat C# nyelven egy NuGet[1] csomag részeként, melyet a fejlesztők egyszerűen be tudnak integrálni a saját tesztkörnyezetükbe. A NuGet csomag használhatóságának demonstrálásához egy konzolos alkalmazás elkészítése. Ezen felül megvizsgálni, hogy az osztási tényező változtatása milyen hatással van a *HDD* algoritmus hatékonyságára. A kiértékelést két különböző, a diszciplína szakirodalmában használatos tesztesethalmazon végezze el, majd az előálló adatokat analizálja különböző mintázatokat keresve. A fő kutatási kérdések, melyeket a hallgatónak meg kell válaszolnia:

- Vajon az osztási tényező általánosítása ugyanolyan hatással van-e a HDD algoritmusra, mint a DDMin-re?
- A HDD során a legjobban teljesítő osztási tényezőt vizsgálva, van-e kapcsolat a tényező és a teszteset mérete között?
- Esetleg az osztási tényező, a tesztesetből előállított fa – vagy annak szintjei – mérete korrelál-e egymással?

Célközönség

- Mérnökinformatikus BSc
- Programtervező informatikus BSc
- Gazdaságinformatikus BSc

Szükséges előismeretek

- Python, C#
- Angol: olvasási szintű készség

[1] https://www.nuget.org/

Adminisztráció

	Külső témavezető	Belső témavezető
Név	Vince Dániel	Dr. Kiss Ákos
Titulus	tudományos segédmunkatárs	adjunktus
Munkahely	Szoftverfejlesztés Tanszék (SZTE)	Szoftverfejlesztés Tanszék (SZTE)
E-mail	vinced@inf.u-szeged.hu	akiss@inf.u-szeged.hu

Feldolgozandó irodalom

[1] Á. Kiss, „Generalizing the Split Factor of the Minimizing Delta Debugging Algorithm," *IEEE Access,* pp. 8:219837-219846, 2020.

[2] R. Hodován és Á. Kiss, „Modernizing Hierarchical Delta Debugging," in *7th International Workshop on Automating Test Case Design, Selection, and Evaluation (A-TEST 2016)*, Seattle, Washington, USA, 2016.

[3] R. Hodován, Á. Kiss és T. Gyimóthy, „Coarse Hierarchical Delta Debugging," in *33rd IEEE International Conference on Software Maintenance and Evolution (ICSME 2017)*, Shanghai, China, 2017.

[4] Á. Kiss, R. Hodován és T. Gyimóthy, „HDDr: A Recursive Variant of the Hierarchical Delta Debugging Algorithm," in *9th ACM SIGSOFT International Workshop on Automating Test Case Design, Selection, and Evaluation (A-TEST 2018)*, Lake Buena Vista, Florida, USA, 2018.

Ütemezés

Félév	Hónap	Feladat leírása
1	1	A feladatkör megismerése, a már implementált programrészek tanulmányozása. A teszthalmazok használatának elsajátítása.
	2	A feldolgozandó irodalom megismerése, konzultáció az esetleges félreértések korai megoldása érdekében. NuGet csomagkezelés megismerése.
	3 4	A módosított algoritmus(ok) implementálása C# programozási nyelven.
2	5	Az implementáció validálása a már létező Picire[2] és Picireny[3] eszközök segítségével a megfelelő teszthalmazokon.
	6	Hibajavítás, az eredmények kiértékelése, minták keresése.
	7	Az eredmények kiértékelése, minták keresése. Konzultáció: a dokumentáció struktúrájának előterjesztése.
	8	Az implementált eszközök, előállt mérési eredmények és konklúziók archiválása. A szakdolgozat dokumentálása.

[2] https://github.com/renatahodovan/picire
[3] https://github.com/renatahodovan/picireny