Java programok hívási gráf alapú vizsgálata

Fejes Endre IV. Inf., fejese@fazekas.hu

Konzulens: Majzik István, BME Méréstechnika és Információs Rendszerek Tanszék, majzik@mit.bme.hu

A statikus programellenőrzés technikái a program futtatása nélkül, a kód elemzésével képesek jellegzetes hibákat felderíteni (pl. elérhetetlen utasítások, túlkomplikált vezérlési struktúra). Az egyszerűbb eszközök a forráskódon végzik el a jellegzetes hibaminták keresését, míg a fejlettebb eszközök a program absztrakt reprezentációi, pl. a vezérlési gráf vagy a hívási gráf alapján is tudnak ellenőrzéseket végezni. A tudományos diákköri dolgozat témája *hívási gráf alapú vizsgálati módszerek* kidolgozása és integrált eszköz formájában történő megvalósítása Java programokhoz. Bár léteznek hasonló funkcionalitású eszközök, a dolgozatban szereplő megoldások több újdonságot is felmutatnak.

Amennyiben a program forráskódja adott, a hívási gráf felépítése a szintaxis elemzésével történik (az Abstract Syntax Tree technológia segítségével). A hívási gráf felépítését a dolgozatban szereplő eszköz kiterjeszti a *bináris formában található komponensekre*, a Javassist technológia segítségével feltérképezve a hívásokat. Így az analízis egységesen tudja kezelni a forráskód formájában nem hozzáférhető komponenseket is. Az eszköz lehetőségei a következők:

- A konkrét hívási gráf alapján a teljes programra közvetlenül származtatja azokat a *komplexitási mértékeket* (pl. hierarchikus komplexitás, hívási szintek száma), amelyekre a biztonságkritikus szoftverek fejlesztési szabványai betartandó határértékeket fogalmaznak meg. Jelenleg a MISRA (Motor Industry Software Reliability Association) szervezet által rögzített releváns mértékek elemzése történik meg.

- A hívási gráf alapján megtalálja a forráskódban azokat a *felesleges kódrészleteket* (osztályokat, metódusokat), amelyek törlésével a programkód jelentősen egyszerűsíthető. Az automatikusan végrehajtható törlés a kódméret csökkentése mellett a karbantarthatóságot is javítja. A kódtisztítás során több érdekes problémát is meg kellett oldani, pl. az implicit hívási pontok meghatározását.

- A programkarbantartás jellegzetes feladata a *hatásanalízis*, tehát annak meghatározása, hogy a programban egy adott helyen történt változtatás mit befolyásol. A hívási gráf alapján az eszköz meghatározza, hogy egy adott metódus módosítása milyen (hívott illetve hívó) metódusok viselkedésére lehet hatással. Ez alapján lehetőség van a regressziós tesztelés mértékének csökkentésére. Ezen kívül a programkód automatikus felműszerezésével az eszköz lehetővé teszi, hogy tesztelés közben a *tényleges hívások követhetők legyenek*, így egyrészt validálva a statikus analízissel meghatározott hívási gráfot, másrészt pedig meghatározva a teszt fedettségi mértékeket.

Az eszköz szorosan integrált az *Eclipse* fejlesztőkörnyezetbe, kihasználva a *Plugin Development Environment* lehetőségeit. Grafikus felületen keresztül valósul meg mind a metrikák és hatásanalízis eredményeinek kijelzése, mind pedig a kódtisztítás indítása. Mivel az eszköz egyszerre képes bináris és forrásállományokat kezelni, széles körben felhasználható.