# Java statikus hívási gráf kinyerő eszközök összehasonlítása

*Ságodi Zoltán*
*II. évf. programtervező informatikus BSc*

*Kóbor Ervin*
*II. évf. programtervező informatikus BSc*

*Témavezetők: Dr. Jász Judit, Pengő Edit, Dr. Siket István*
*SZTE TTIK Szoftverfejlesztés Tanszék*

A hívási gráfok (*Call Graphs*), amelyek csúcspontként metódusokat, élként pedig metódusok közti hívásokat tartalmaznak, alapvető elemei a fejlett interprocedurális vezérlésátadást és adatfolyamot elemző feladatoknak. Két főbb változatuk létezik attól függően, hogy milyen alkalmazást szeretnénk általuk támogatni: a dinamikus és a statikus hívási gráfok. A dinamikus hívási gráfok építésekor a kód futtatása szükséges, lehetőleg úgy, hogy a program lefedettsége a lehető legnagyobb legyen. A gráf építése közben nemcsak maga a hívási gráf, de az egyes hívási útvonalak vizsgálata is segítheti egy-egy alkalmazott algoritmus működését. Statikus hívási gráfok építésekor ugyanakkor nem kell futtatnunk a kódot, nincs szükség tesztesetekre, egyszerűen a forráskód alapján építjük meg a lehetséges hívásokat tartalmazó gráfot.

A Java nyelvez hasonló dinamikus programozási nyelveknél, melyek lehetőséget adnak többalakúságra (polimorizmus), reflekcióra (futásidőben betöltött programrész) a statikus hívási gráfok között a tekintetben is lehet eltérés, hogy milyen éleket kötnek be egy-egy szituációban. Elképzelhető, hogy különböző hívási gráfot építő algoritmusok más és más éleket kötnek be a felhasználásuktól függően. Nem biztos, hogy a különböző hívási gráfok között megállapítható, hogy melyik szolgáltatja a legjobb eredményt, kiválasztásuk mindig függ a további felhasználásuktól. A gyakorlatban azonban nemcsak az egyes hívási gráfépítő algoritmusoktól függ egy-egy eszköz kimenete. Nagyon sokszor a különböző nyelvi elemeket is máshogy kezelik le, amelyek azonban nem feltétlen szolgálják a kapcsolódó alkalmazások működését.

A dolgozatban hat statikus, Java források hívási gráfjának előállítására alkalmas nyílt forráskódú programot (OpenStaticAnalyzer, Soot, Spoon, Caller Hierarchy Printer, WALA, Gousiosg) mutatunk be és hasonlítunk össze. Ezekkel a programokkal Java nyelven íródott kódokat elemeztünk, illetve hasonlítottuk össze a kapott hívási gráfokat. Az összehasonlítás során saját programot készítettünk, mely képes párosítani a fenti eszközök által előállított gráfok csomópontjait és éleit, ezzel összehasonlíthatóvá téve a gráfokat. A dolgozatban bemutatásra kerül az, hogy ez a párosítás nem triviális feladat, illetve hogy a felmerülő problémákat hogyan kezeltük. Emellett a programokat három nagyobb nyílt forráskódú rendszeren futtattuk (méretük 40 és 200 ezer sor között változott), és egy saját magunk által készített példán (amely a legtöbb Java 8 által támogatott hívási lehetőséget tartalmazta), hogy bemutassuk, mekkora és milyen típusú eltérések lehetnek az egyes eszközök között a gyakorlatban.

A kapott eredmények segítségével megmutatjuk, mik azok a nyelvi jellemzők, amelyek legnagyobb mértékben módosíthatják a statikus hívási gráfokat, illetve megmutatjuk ezek előfordulási gyakoriságát is a nagyobb példák elemzése által. Az eredmények segíthetik egy-egy hívási gráfra épülő alkalmazás készítőit abban, hogy milyen hívási gráfot előállító eszközt válasszanak, illetve ha saját maguk építik fel ezt a gráfot, mik azok a nyelvi elemek, amelyek vizsgálatára különleges hangsúlyt kell helyezniük.