Önálló laboratórium dokumentáció

Kivonat – Lucz Tamás Soma, Méréstechnika és Információs Rendszerek Tanszék

Összetett szoftverek fejlesztése során a kódbázis növekedésével általában a kódban megjelenő fejlesztői hibák száma is nő. Ezen hibák fokozott kockázatot jelenthetnek, hiszen az esetlegesen helytelen, nemkívánatos működés mellett jelentős biztonsági réseket eredményezhetnek. Kiaknázásuk által rosszindulatú támadók a szoftvert számukra kedvező, az eredetileg tervezettől eltérő módon futtathatják.

A statikus forráskódanalízis egy, az iparban gyakran használt, általánosan elfogadott szoftvertesztelési megközelítés. Célja, hogy minél több szoftverfejlesztői hibát minél előbb, még a program fejlesztési szakaszában – a kód lefordítása és lefuttatása nélkül – tárjon fel, csökkentve ezzel a működés közben felmerülő programhibák számát, és így a telepítés utáni hibajavítással járó pluszköltségeket. Felhasználási lehetőségei közé tartozik a csoportos, vállalati kódolási szabályoknak, stílusoknak való megfelelés ellenőrzése, illetve egyre több statikus analízis eszköz nyújt támogatást egyre komolyabb logikai hibák fordítási vagy akár kódírási idejű feltárásához is.

Napjaink folytonos integrációs infrastruktúrájába illesztve a statikus analízis hatékony eszköz lehet a fejlesztői hibák feltárásában, és ezáltal az állandó kódminőség biztosításában. Nagymértékű népszerűsége ellenére a JavaScript nyelvhez – annak dinamikus és gyenge típusosságából eredő sajátosságok következményeként – kevés statikus analízis-eszköztár létezik, és a rendelkezésre álló eszközök sem nyújtanak teljeskörű megoldást nagyméretű, vállalati szintű JavaScript forráskódtárak összefüggő elemzéséhez. Gyakran felmerülő probléma emellett az analitikus komplexitással általában fordítottan arányos sebesség: sem folytonos integrációs infrastruktúrába, sem fejlesztőkörnyezetbe nem illeszthető olyan eszköz, amely miatt a fordítási idő akár órákkal növekszik.

A szakdolgozatom egy már létező, a fenti követelményeknek nagy részben eleget tevő statikus kódanalízis-keretrendszer bővítésének megtervezését, implementációját, illetve tesztelését és értékelését kísérte figyelemmel.

Az önálló laboratóriumom során a fenti, gráfadatbázisra épülő statikus kódanalízis-rendszert fejlesztettem tovább. Munkám jelentős részében a rendszer alatt működő JavaScript-parsert egészítettem ki úgy, hogy támogassa a legújabb JavaScript szabványt, az ES2017-et. Leginkább a szabvány fő újítása, az aszinkronitás támogatása hozott újdonságokat, amik miatt jelentősen át kellett alakítani a parsing szemantikáját.

Másrészt, mivel a kódbázisok első gráfadatbázisba történő importálása és átalakítása – amikor inkrementális feldolgozás még nem lehetséges – túlságosan lassú volt ahhoz, hogy valódi, nagyméretű projekteken használni lehessen, szükség volt egy új módszerre, amellyel a folyamaton gyorsítani lehet. Itt egy CSV-alapú megoldást alkalmaztam: közvetlenül CSV-fájlból gráfadatbázist generálva 10-szeres gyorsulást sikerült elérni.