Kivonat

Nagyméretű, komplex szoftverek fejlesztésekor a közreműködők számának növekedésével a kódban is gyakrabban fordulhatnak elő hibák. Ezek kiküszöbölésére jelenleg is rendelkezésre állnak megoldások, mint például a statikus analízis, amely többek között automatikusan ellenőrzi, hogy a kód megfelel-e a kódolási szabályoknak.

A statikus analízis a gyakorlatban sokszor lassú és drága művelet. Különösen folytonos integrációs alkalmazása során jelent skálázhatósági kihívást, mert minden módosítás után újra kell futtatni az ellenőrzést a teljes kódbázison. Ugyanakkor nagy igény jelentkezik a statikus analízis technológiák gyorsítására, mivel a gyakorlatban a tesztelést és így a fejlesztés egészét jelentősen gyorsíthatja, hiszen segíti a hibák korai szűrését.

Az egyik lehetséges megoldás az inkrementális feldolgozás. A TDK dolgozat keretében olyan rendszert készítettem, mely segítségével a felhasználók által definiált problémákra magas szinten, gráf minták alapján lehet keresni. A rendszer inkrementalitásának lényege, hogy a lekérdezések kiértékelése és riport első generálása után a rendszer hatékonyan fel tudja dolgozni a kód változásait is, így a későbbi futások jóval hatékonyabbak.

A rendszer működőképességének igazolására olyan méréseket terveztem, melyek nyílt forrású programkódok analízisfolyamatainak végrehajtásával betekintést nyújtanak a rendszer skálázhatóságára az elemzett kódbázis méretének tükrében. Ezeket a méréseket elosztott környezetben végeztem el.

Abstract

In large-scale complex software development, the number of coding errors are noticeably increasing with the number of contributors. Mitigatory solutions for this problem exist, but they have their limitations. For example, static analysis methods that verify that code is compliant with coding conventions are frequently very resource intensive in practice. Especially for continuous integration purposes, the size of the codebase would require a scalable solution because for every changeset in the source, the entire verification process needs to be reapplied on the whole codebase.

Incremental processing is one of the possible solutions for speeding up static analysis. In my report I present a system that is able to search for coding problems based on high level specifications by graph patterns. The system supports incremental evaluation by processing source code changes and mapping them efficiently to stages of the processing workflow. Hence, after the initial query evaluation and report generation, consecutive runs are significantly more efficient.

To investigate the scalability of the described system, I present benchmark results that are based on measurements carried out with open source code bases and standard coding validation rules. These benchmarks were executed in a distributed environment.