Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Kódlefedettség mérése .NET környezetben

Molnár Gábor (HSVKEF), I. évf, (MSc) mérnök inf. szakos hallgató
Konzulens: dr. Micskei Zoltán adjunktus, MIT
Szolgáltatásbiztos rendszertervezés szakirány
Önálló laboratórium 1. összefoglaló
2013/14. II. félév

A kódlefedettséget (code coverage) több szinten is mérhetjük. Az egyik legegyszerűbb lefedettségi metrika az utasítások fedettségét ellenőrzi (statement coverage). Ezt jellemzően az összes elérhető kódlefedettség vizsgáló eszköz képes mérni, így például többek között a Visual Studio is képes mérni. Bizonyos esetekben ennek a használata nem kifizetődő, így például egy build szerveren vélhetően egy nyíltforrású, ingyenes eszközt szeretnénk üzemeltetni, illetve egy bonyolultabb metrika, az elágazásfedés (branch coverage) vizsgálata is érdekes a fejlesztés szempontjából.

Utóbbi mérése már kevés eszközben elérhető, ilyen például a licencköteles NCover, illetve bizonyos mértékben az ingyenes OpenCover (az eredmények megjelenítéséért felelős ReportGeneratorral kiegészítve).

A fejlesztési folyamathoz szorosan kapcsolódik egy folytonos integrációs kiszolgáló (CI) alkalmazása is, itt a különböző eszközöket általában egy plugin formájában célszerű használni, mivel így egyszerűbb a konfigurálás, mint parancssorból futtatva, és elkerülhető a közös adatok duplikálása (például a futtatható fájlok elérési útja).

Az önálló laboratóriumi munkám két részből állt:

- Mivel a ReportGenerator jelenlegi verziója nem jeleníti meg a nem fedett elágazásokat, így az egyik feladatom ennek a funkciónak az implementálása.
- A Jenkins alapú folytonos integráció megvalósítása érdekében megvizsgálom a pluginfejlesztés lehetőségeit, és elkészítem az OpenCover és a ReportGeneratort meghívó kiegészítőket.

Az OpenCover kimenetében a *SequencePointok* azonosítják az utasítás sorokat, ezekhez meg is jelenik a forrássor információ, azonban az elágazásoknak megfelelő *BranchPointokat* csupán a metóduson belüli IL (köztes kód) ofszetje azonosítja. A feladat tehát ennek a köztes kódnak a forrás sorokra képzése. Ehhez a Mono.Cecil könyvtárat és a PDB fájlokat használtam. Mint kiderült a PDB fájlokból nem olvasható ki sorinformáció minden IL ofszethez, így a módszer nem minden esetben használható.

Egy másik megközelítés a forráskód irányából történt. Amennyiben, minden forrássorhoz meg tudjuk határozni azokat az IL utasításokat, amelyek a sor fordításából adódnak, akkor ismerjük azokat az utasításokat is, amik fedése szükséges. Ehhez a Microsoft Compiler Platformot használtam (Rosyln API). Ez a megközelítés (jelenleg) nem teszi lehetővé az IL ofszet meghatározását, így ez sem használható. Ellenben, mintegy mellékhatásként, megismertem a Roslyn API szintaxisfa alapú szemantikus analízisét is.

A Jenkins plugin fejlesztése egy Maven alapú, jenkins-es archetípusú projekt létrehozásával kezdődött, ezt követően a definiált *ExtensionPoint* osztályokból leszármazással készítettem el a plugint. A pluginek alapvetően háromfélék lehetnek: foglalkozhatnak verziókezeléssel, egy adott fordítási lépéssel, illetve az eredmények publikálásával. A megvalósított ReportGenerator plugin-hoz az utolsót használtam (*Recorder*).

A továbbiakban az elágazásfedés mérésének (mélyebb) megértéséhez az OpenCover forráskódjának megismerése szükséges, ezt követően lehetséges úgy módosítani az OpenCovert, hogy forrássor-információkat tároljon a kimeneti XML fájlban, illetve ez megvalósítható a ReportGeneratorban is.

A pluginfejlesztéssel kapcsolatosan az OpenCovert futtató *Builder* kiegészítés fejlesztése is lehet a további munka alapja, illetve esetlegesen a közös parancshívással foglalkozó logika is általánosítható.