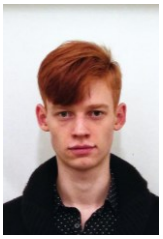
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Automatikus build végrehajtás



Garda Zsolt Barnabás III. évf, (BSc) mérnök inf. szakos hallgató
Konzulensek: Monostori Dénes, ThyssenKrupp Presta Hungary Kft,
Dr. Majzik István egyetemi docens, MIT
Informatikai technológiák szakirány / Rendszertervezés ágazat
BSC Önálló laboratórium összefoglaló
2015/16. II. félév

A szoftverek fejlesztése napjainkban egyre nagyobb kihívást jelent: az egyre több funkcionalitással bíró szoftverek egyre nagyobb komplexitásúak, a fejlesztések során korábban megírt kódok felhasználása egyre hangsúlyosabb, ha gyorsabban szeretnénk eredményeket elérni.

A fejlesztőknek minden segítség fontos, hogy versenyben tudjanak maradni az idővel és a versenytársaikkal, minőségi alkalmazásokat tudjanak készíteni minél kevesebb olyan részletre való odafigyeléssel, ami amúgy automatizálható. Az automatizálással több hibaforrás is kiküszöbölhető, a minél sűrűbben való teszteléssel pedig jobban biztosítható a termék időre való elkészülése.

Ezekre a feladatokra használhatóak a Continuous Integration rendszerek, amelyek egységbe foglalják a szoftver forráskódjától a végtermék előállításáig előforduló lépéseket, úgy, mint a kódok verziókezelőből való letöltése, azok buildelése akár több lépésen keresztül, majd az elkészült terméken tesztek lefuttatása, kódminőségi metrikák mérése, és az eredmények vizuálisan, egységesen való prezentálása, hogy minél gyorsabban megtudjuk a build eredményét.

Az Eclipse RCP alkalmazások fejlesztése sok előnnyel jár, hiszen a moduláris felépítésük miatt felhasználhatunk már több olyan egységet, amit mások elkészítettek. Éppen ezért azonban ha változtatás történik egy ilyen modulban, azt a változást nekünk is követnünk kell, ami hibaforrást jelent. Ezért is lenne célszerű, ha az RCP alkalmazásokat Continuous Integration környezetben állíthatnánk össze.

A Countinuous Integration rendszerek (mint például a Hudson és Jenkins) erősen támogatják a Maven build tool-t az alkalmazás build folyamatának végrehajtásához, és a Maven egyre elterjedtebb, hiszen hasznos szolgáltatásokat nyújt. Azonban az Eclipse RCP alkalmazások Mavennel való buildelése problémákat vet fel.

Ennek megoldására a Tycho nevű Maven plug-in nyújt segítséget, aminek a beüzemelése azonban koránt sem egyszerű feladat főképp a hiányos dokumentáció miatt.

A félév folyamán megismerkedtem az említett technológiákkal és kidolgoztam egy lépéssorozatot, amelynek segítségével a legtöbb Eclipse RCP alkalmazás forrásait át tudjuk konvertálni úgy, hogy az a Maven segítségével buildelhető legyen.

Először egy kisebb RCP alkalmazást alakítottam át, majd egy több plug-in-ból, feature-ből és JUnit tesztesetből álló alkalmazást. Kitapasztaltam sok hibalehetőséget, a konfiguráció során elvégzendő lépéseket, megadandó paramétereket pedig összefoglaltam a félév végi dokumentációmban.

Hozzáadtam a folyamathoz egy szolgáltatást, ami az egyes Eclipse plug-in és feature projektekben a Maven buildhez szükséges POM.xml állományokat automatikusan kigenerálja. A build során képes az átalakított rendszer P2 Update Site-okat is létrehozni az alkalmazás mások számára elérhetővé tételéhez.

Az egész build folyamatot Continuous Integration környezetbe helyeztem, az alkalmazás buildelése és a JUnit tesztek eredményeinek grafikus megjelenítése egy gombnyomásra működik.

A továbbiakban szeretném a Continuous Integration környezet adottságait jobban kihasználni a SonarQube kódminőségi metrikákat mérő Jenkins plug-in hozzáadásával illetve az RCPTT GUI teszteket futtató rendszer folyamatba helyezésével.