Teszthatékonyság vizsgálata beágyazott rendszerekben

Önálló laboratórium feladat összefoglalója (1. félév)

Répás Gergő (HCNRVV)
Konzulens: Pintér Gergely
BME Méréstechnika és Információs Rendszerek Tanszék
Informatikai infrastruktúra tervezése szakirány, 2005/2006. I. félév

A tárgy keretein belül szoftvertesztelés hatékonyságának mérésével foglalkoztam. Teszthatékonyság mérésére szolgálnak a következő mérőszámok: utasítás lefedettség, döntési ág lefedettség, feltétel lefedettség és út lefedettség. Ezek közül az utasítás lefedettséggel foglalkoztam, mely a tesztelés során végrehajtott utasítások számának és az összes utasítás számának arányát adja meg.

A feladatom egy olyan utasítás lefedettséget mérő eszköz megvalósítása volt, amely biztonságkritikus, valósidejű beágyazott rendszerekre írt C nyelvű szoftverek utasítás lefedettségét méri. A megvalósított megoldás a forráskódot műszerezi fel olyan módon, hogy az utasítások végrehajtását a memóriában foglalt tömbbe naplózza. A teszt lefuttatása után a bejárási információ kiolvasható, majd a végre nem hajtott utasítások színezéssel kiemelhetők a forráskódban. A kialakítást a beágyazott rendszerek adottságaihoz kellett igazítani, azaz alacsony erőforrás használat mellett kell megoldani a feladatot.

A feladat megoldását Java nyelven végeztem, a grafikus megjelenítést Eclipse szerkesztőablakként írtam meg.

Az első eszköz, amit készítettem, a C forrás felműszerezésére való parancssoros eszköz. Ennek működése a következő három fázisból áll:

- Első fázis: a C forrás nyelvi elemzése és az absztrakt szintakszisfa felépítése. A nyelvi elemző elkészítéséhez a JavaCC nevű eszközt választottam, melyhez egy szabadon elérhető ANSI C nyelvtan is rendelkezésre áll. A nyelvi elemzés során az absztrakt szintakszisfa XML reprezentációja készül el.
- Második fázis: az absztrakt szintakszisfa elemzése és a naplózó utasítások elhelyezése. Először a szekvenciálisan végrehajtódó utasításblokkok felismerése történik, és ezen utasításblokkok végére helyezem el a bejárási információt létrehozó utasításokat. Ennek megvalósítása XSLT-vel történik. Ebben a szakaszban jön létre az a fájl is, amely az utasítás-blokkok azonosítójához tartozó pozíció-információkat tartalmazza.
- Harmadik fázis: kódgenerálás. Az XML reprezentációból ezek után elkészül a generált kód.

Ezt az eszközt integráltam az Eclipse CDT nevű C/C++ fejlesztőkörnyezetbe, mely eredményeképp az alapértelmezett „Release” és „Debug” fordítási mód mellet egy felműszerezett binárist előállító mód is megjelenik.

A második eszköz a végre nem hajtott utasítások színezésére való. Ez egy Eclipse szerkesztő, melynek három bemenete van: az eredeti forráskód, a futás során kialakult bejárási információ és az utasítás blokkok azonosítójához tartozó pozíció-információ.

Az elkészült eszközökön a következő dolgokat javaslom fejleszteni:

- A nyelvi elemző ne csak ANSI C-t, hanem a GCC C dialektusát is fogadja el. Esetleg a JavaCC helyett az ANTLR nevű nyelvi elemző generátort lehetne használni, amihez elérhető egy ingyenes GCC nyelvtan.
- Az eszköz legyen képes döntési ág lefedettség mérésére.