# Modell alapú robosztusság tesztelés

## Önálló laboratórium feladat összefoglalója (8.félév)

**Kárer Győző (XFICP2)**
**Konzulens: Dr. Majzik István**
**BME Méréstechnika és Információs Rendszerek Tanszék**
**Informatikai infrastruktúra tervezése szakirány, 2006/2007. II. félév**

A különböző számítástechnikai alkalmazások fejlesztésének igen fontos része az alkalmazások tesztelése. A tesztelés segítségével az alkalmazásokban meglévő hibák felderítése a cél. Ez a gyakorlatban általában kézi úton történik és ezáltal a fejlesztési idő jelentős részét lefedi. Általános cél, hogy ezt a teszteléssel töltött időt lerövidítsük, például a tesztgenerálás, futtatás és kiértékelés valamilyen szintű automatizálásával.

A hibák keresése szempontjából többféle típusú hibáról beszélhetünk. Robosztusság hibáról beszélhetünk, ha az alkalmazás extrém paramétereket tartalmazó teszthívások illetve szélsőséges környezeti helyzet (pl. túlterhelés) esetén nem-specifikált működést mutat. Például operációs rendszerek vizsgálata kapcsán tipikus robosztussági hibák lehetnek a következők: (1) *abort* jellegű hiba, amely azt jelenti, hogy a rendszer a teszthívás után is működik, de egy meghatározott időn belül nem kapott választ, (2) *restart* hibák esetében a rendszer újraindult, (3) *crash* jellegű hibák pedig a rendszer összeomlását eredményezik. Az önálló laboratórium feladat keretében **modell alapú robosztusság tesztelés** a feladatom.

A feladat elkészítésének első fázisaként az idevonatkozó szakirodalom segítségével megismertem a területhez tartozó projekteket és az általuk megvalósított technikákat.

A tématerület megismerése után következett a megvalósítandó rendszer tervezése. Az automatikus tesztkészlet generáláshoz kétféle modellt használunk.

- Az *osztály- és objektum diagramok* segítségével a rendszer architektúrájáról kaphatunk információkat. Ezek segítségével megismerhetőek az adott objektumokhoz tartozó metódusok és azok bemeneti paraméterei. A tesztkészlet generálása során kritikus pontként jelenik meg a szélsőséges környezeti helyzetek kezelése. Ezt extrém bemeneti paraméterek megadásával valósíthatjuk meg, amelyek a paraméterek típusához tartozó szélső értékeket illetve azon kívüli (pl. érvénytelen) értékeket jelentik. További fontos kérdés az összetett típusok és az öröklés kezelése is.
- A másik alkalmazandó modell a *szekvencia diagram*. Ennek segítségével, a környezettel történő interakciót vizsgálhatjuk. A szekvencia diagramból kinyert információn mutációkat végezhetünk, ezzel modellezendők azok az esetek (pl. programozói hibák), amikor a szekvenciából kihagynak bizonyos metódusokat, felcserélnek metódusokat, illetve téves hívásként olyan metódushívások jelennek meg amik nem is az adott osztályhoz tartoznak.

A létrehozott tesztkészlettel elvégzett tesztelés eredményeit LTL követelmények alapján egy elfogadó automata segítségével értékelhetjük ki.

A félév utolsó részében a megtervezett robosztusság teszt generáló rendszer implementálásával foglalkoztam, ahol a modelleket az Omondo EclipseUMLEditor segítségével hoztam létre. A megvalósított teszt generátor alkalmazás jelenleg az objektum diagram feldolgozását végzi el és az abban lévő egyszerű típusokhoz készíti el a tesztkészletet. Az extrém értékek hozzárendelése egy konfigurációs fájl segítségével történik.

További célok a teljes tesztkészlet generálása, a szekvencia diagram mutációinak kidolgozása, az LTL alapú elfogadó automata létrehozása (generálása), valamint a generált tesztkészlettel robosztusság tesztelések végzése.