Eclipse plugin for FindBugs és Microsoft Visual Studio Code Analysis eszközök összehasonlítása

BUDAI PÉTER

(budai@iit.bme.hu)

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

2011. 12. 01.

Áttekintés

- Cél a lehetséges programozói hibák detektálása
  - A fejlesztési folyamat minél korábbi pontján
    - Ezzel időt és költséget spórolhatunk
- Statikus analízis
  - A kód lefuttatása nélkül próbáljuk felismerni a potenciális hibaforrásokat
  - Az analízis kiterjedhet
    - Forráskódra
    - Lefordított tárgykódra
      - Ez utóbbi a jellemző (egyszerűbb)
      - Köztes kódot alkalmazó nyelveknél (Java/.NET) még könnyebb dolgunk van, a tárgykód "beszédes"

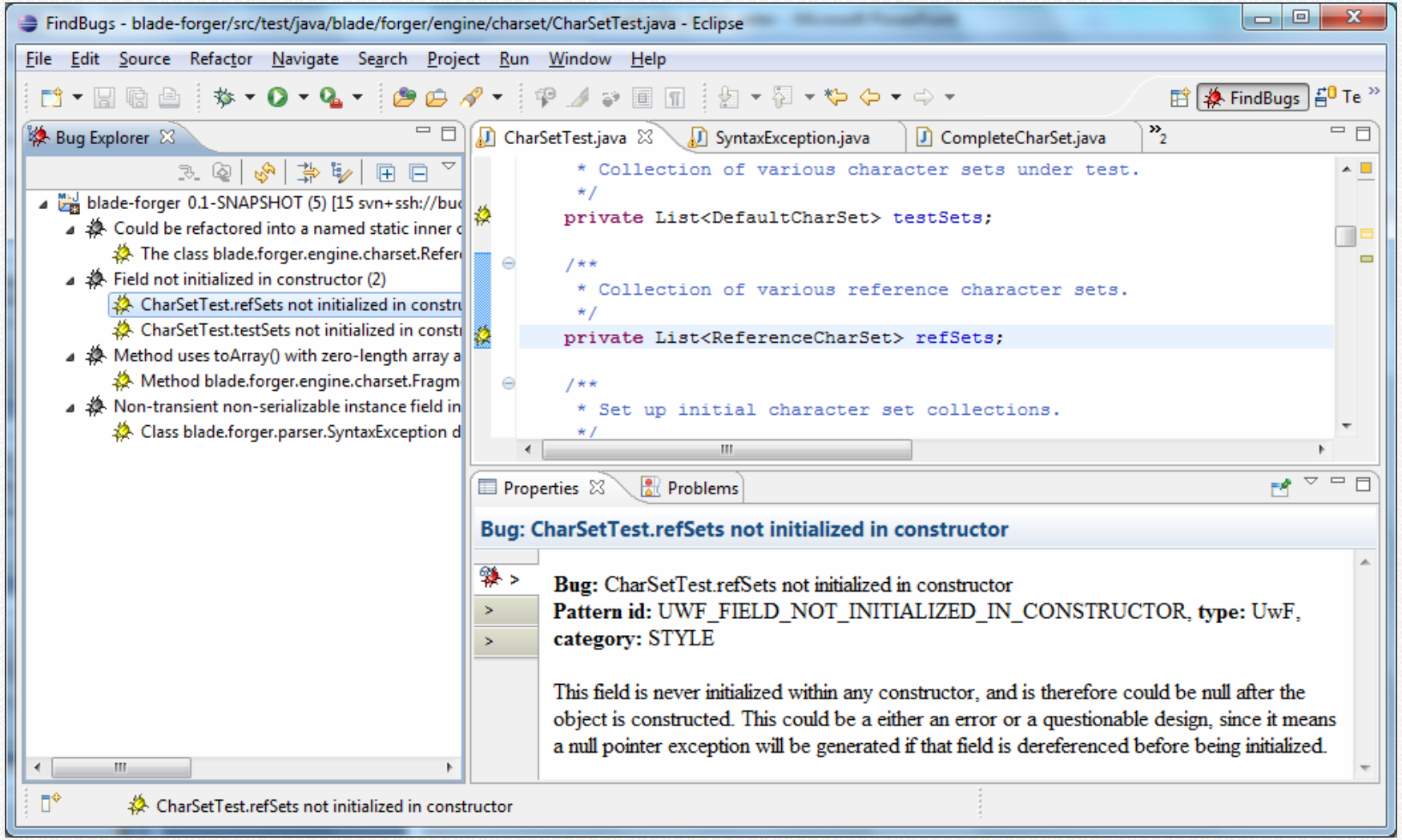
FindBugs

- University of Maryland
  - William Pugh & David Hovemeyer, 2003
  - Nyílt forráskód (LGPL)
- Java nyelvhez (Java SE 1.5+) készült
  - Hibakeresés és helytelen programozói gyakorlatok kiszűrése
  - Hasonló eszközök még:
    - PMD, JLint, (Bandera), (ESC/Java)
- Többféle UI
  - Swing alapú önálló alkalmazás
  - Eclipse plugin
  - Ant task, parancssor, …

FindBugs

- Hogyan működik?
  - .class fájlok statikus analízise (BCEL vagy ASM library)
  - Különböző hibamintákat (bug pattern) keres
    - Előfordulhatnak hamis pozitív esetek (<50%)
    - Könnyen bővíthető újabb mintával
  - Az összegyűjtött hibalistát valamilyen módon visszakapjuk
    - XML állomány
    - Felhasználói felület: Hibalista és forráskód nézet
      - Ez utóbbi feltétele, hogy hozzáférhető legyen a forráskód
  - Szöveges leírások ez egyes hibákhoz

FindBugs Eclipse plugin



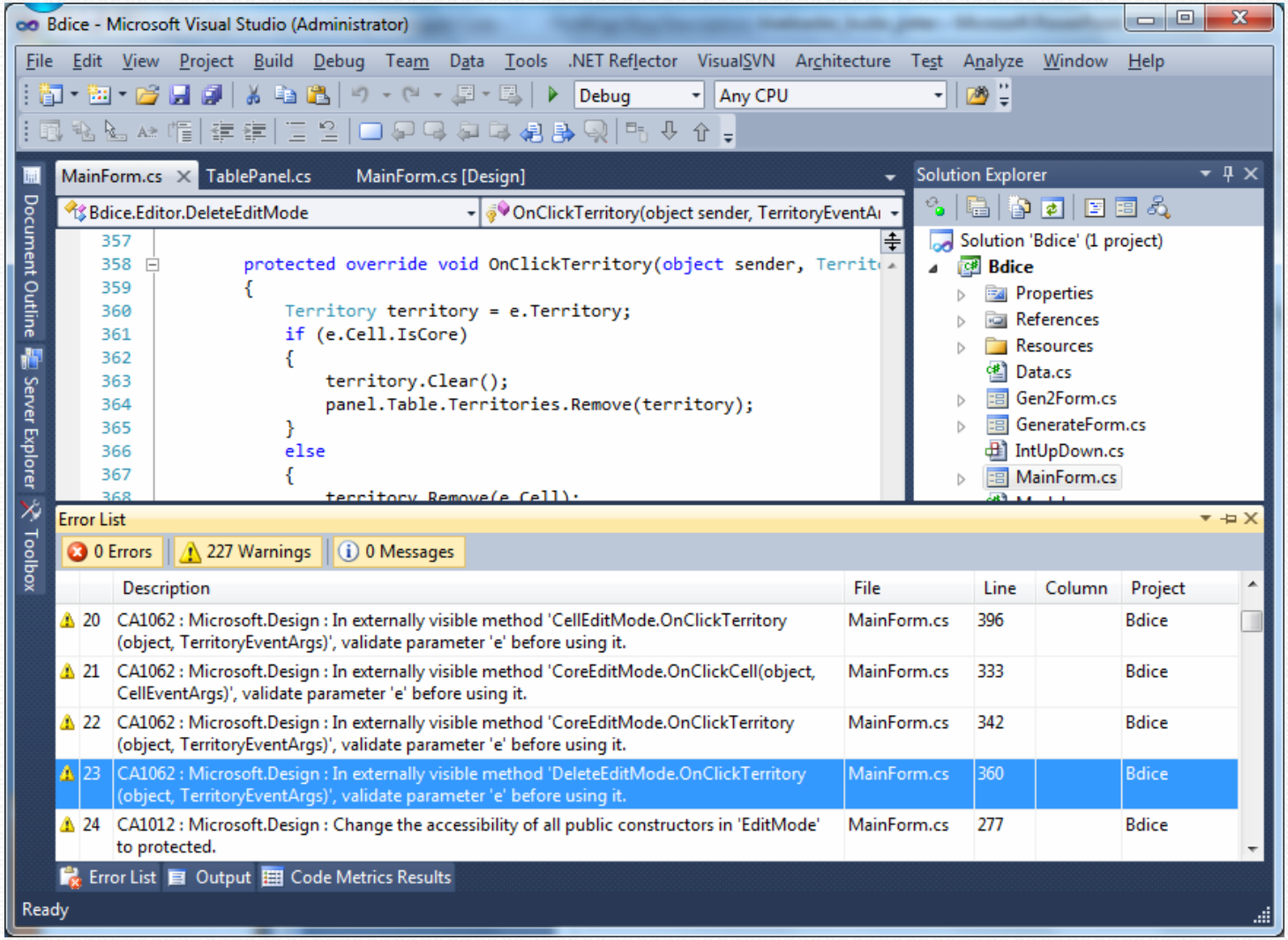
Microsoft Code Analysis

- Microsoft
  - Visual Studio 2010 része
    - Ultimate és Premium változatokban
    - FxCop utódja
  - Nem ingyenes, de MSDNAA-ről elérhető
- C/C++ és bármilyen .NET nyelvhez
  - Hibák felfedezésére és a forráskód minőségének javítására
  - Hasonló eszközök:
    - Gendarme, StyleCop, dotTEST
- Visual Studio-ba integrált kezelőfelület

Microsoft Code Analysis

- Hogyan működik
  - CIL köztes nyelvű szerelvények statikus analízise
  - Szintén hibamintákkal dolgozik
    - Bővíthető ugyan, de nem támogatott
    - Az egyes hibajelzések attribútumokkal kikapcsolhatók a kódból
  - A felismert hibák a fejlesztőkörnyezetben jelennek meg
  - Szöveges leírások az egyes hibákhoz
    - Bővebb ismertetők
    - Tippek javítási lehetőségekre

Microsoft Code Analysis



Összehasonlítás

- Mindkét eszköz hasonló elven működik
  - Köztes nyelvű kódon dolgoznak
    - Java bytecode ⇔ .NET IL
  - Hibamintákat keresnek
    - A különbség az ismert minták számában van:
      - 548 (FindBugs) ⇔ 214 (CodeAnalysis)
    - Ez egyes minták vizsgálata ki/be kapcsolhatók
      - Vannak csak költségesen ellenőrizhető minták
    - Bővíthetőek újabb mintákkal
      - Csak a szükséges erőfeszítés mértéke eltérő
- Integráció a fejlesztőkörnyezetbe

Összehasonlítás

- Hibaminták csoportosíthatók
  - Súlyosság szerint
    - FindBugs: Low, Medium, High prioritás
    - CodeAnalysis: Ignore, Warning, Error
  - Hiba típusa szerint
    - FindBugs:
      - Bad practice, Correctness, Multithreaded correctness, Malicious code vulnerability, *Internationalization*, *Performance*, *Security*, Dodgy code
    - CodeAnalysis:
      - Design, *Globalization*, Interoperability, Maintainability, Mobility, Naming, *Performance*, Portability, Reliability, *Security*, Usage

Hibakategóriák

- FindBugs
  - Internationalization
  - Bad practice
  - Performance
  - Malicious code vulnerab.
  - Correctness
  - Multithreaded correctn.
  - Security
  - Dodgy code
- CodeAnalysis
  - Globalization
  - Design
  - Naming
  - Usage
  - Maintainability
  - Reliability
  - Security
  - Mobility
  - Interoperability
  - Portability
  - Performance

Hibakategóriák

- FindBugs
  - Internationalization (2)
    - String-byte konverzió
  - Bad practice (86)
    - equals() és társai
    - finalize()
    - API konvenciók
    - Elnevezési konvenciók
    - I/O, szerializáció
  - Performance (61)
    - Boxing
    - String művetek
- CodeAnalysis
  - Globalization (11)
    - Alkalmazás-lokalizáció
  - Design (62)
    - API konvenciók
    - Névtér, interfész és metódus tervezési kritériumok
  - Naming (25)
    - Pozitív és negatív elnevezési konvenciók
  - Usage (44)
    - Dispose, finalizer
    - Szerializáció
    - Operátor felüldefiniálás

Hibakategóriák

- FindBugs
  - Malicious code vulnerab. (14)
    - Fokozott adatrejtés
  - Correctness (213)
    - Rendkívül változatos
    - Bitműveletek
    - equals()
    - null
    - Dead store, visszatérési érték
  - Multithreaded correctn. (52)
    - Szinkronizáció helytelen használata
    - Nem szálbiztos hívások
- CodeAnalysis
  - Maintainability (6)
    - Kód komplexitás
  - Reliability (6)
    - Natív erőforrások
  - Security (44)
    - CLR-specifikus biztonsági elemek
  - Mobility (2)
    - Energiagazdálkodás

Hibakategóriák

- FindBugs
  - Security (9)
    - SQL és HTTP sebezhetőségek
  - Dodgy code (92)
    - Felesleges, ismétlődő kódrészletek
    - Elérhetetlen kód, változó, metódus
- CodeAnalysis
  - Interoperability (15)
    - COM és Win32
  - Portability (3)
    - OS és .NET kompatibilitás
  - Performance (17)
    - Használaton kívüli elemek

Konklúzió

- Jól használható eszközök
- A hibaminták között aránylag kicsi az átfedés
  - Nem várt eredmény
  - Sok a platformfüggő elem
  - Mindkettőben előfordul:
    - Nem használt kód felderítése
    - Az Object örökölt metódusainak működése
    - Elnevezési konvenciók

Konklúzió

- Kicsit más megközelítések
  - FindBugs
    - Több, aprólékosabb szabály
    - A kód helyessége, hibalehetőségek felderítése hangsúlyos
  - CodeAnalysis
    - Kevesebb, átfogóbb, magasabb szintű szabályok
    - Több a nyelvi és platform konvenciókat ellenőrző minta
  - Megfigyelhető az akadémiai és az ipari világ közti különbség
- Statikus analízis önmagában nem elegendő

Kérdések