Jak testovat software v praxi

aneb

šetříme svůj vlastní čas

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Proč testy nepíšeme

- Nemáme na to čas
  - Platí v cca 5% případů
    - Nový projekt
    - Prototyp je třeba mít během pár dní
    - Počítá se s tím, že další verze se napíše znovu
- Zákazník to nezaplatí
  - Je třeba mu vysvětlit, že ušetří na údržbě a na budoucím rozvoji
  - Zahrnout to do nákladů na vývoj
- Není to potřeba
  - Legitimní u malých krátkodobých projektů bez budoucího rozvoje

Proč testy psát

- Investujme trochu času teď
- V budoucnu se nám mnohonásobně vrátí

- Některé výhody
  - Nerozbijeme to, co už fungovalo a bylo otestované
  - Programátoři se nebudou bát opravovat chyby
  - Neuděláme hloupé chyby
  - Chyby se nám nebudou vracet

Úrovně testování

- Developer testing
  - Zmáčknout F5, proklikat, vyzkoušet hned
  - „Debugovací“ výpisy, dočasný kód
  - Nejlevnější a nejrychlejší oprava chyby

  - Výhoda: dělá každý vývojář (výjimky zastřelit!)
  - Problém: opakovatelnost
    - Převést Debug.WriteLine a ruční kontrolu na automatizovaný test je práce na 2 minuty!

Test z dočasného testovacího kódu

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Úrovně testování

- Unit testing
  - Testování izolovaných jednotek
  - Vhodné mimo jiné
    - Pro různé algoritmy
    - Vlastní kolekce
    - Kusy kódu bez závislostí
  - Mnoho frameworků – NUnit, xUnit, MS Test

Vlastnosti unit testů

- Nezávislé na sobě
- Nezávislé na pořadí spouštění
- Paralelizovatelné (ideálně)
- Nezávisí na stavu prostředí
- Izolace od závislostí
  - Mockování
    - Náhrada reálné funkcionality zjednodušeným mechanismem pro účely testování
      - Platební brána, rozesílání e-mailů, místo DB kolekce v paměti, odstínění od filesystému apod.

Unit testy a jejich parametrizace

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Na co si dát při testech pozor

- Extrémní případy
  - Číslo v poli je minimum (maximum)
- Chybné vstupy
  - Ale není nutno všude, nebrat dogmaticky
  - Psát unit test na každý if…throw je ztráta času

Úrovně testování

- Integrační testy
  - Testují spolupráci více komponent systému
  - V praxi jich napíšete zřejmě mnohem víc než unit testů
    - Záleží na projektu
  - Blíže realitě – nesnažíme se o kompletní izolaci
    - Typicky jdeme proti reálné databázi

Testování Entity Framework modelu

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Testování proti reálné DB

- Testy by neměly databázi měnit
  - Opakovatelnost, determinismus
  - Možnosti
    - Před testem zahájit transakci a na konci rollback
      - Nelze použít vždy
    - Před testem udělat RESTORE databáze
      - Pomalé
    - Testy po sobě uklidí samy
    - Dobře řešeno má např. RavenDB
      - Umí pracovat v inmemory módu

Testování proti reálné DB

- Typicky separátní databáze s předpřipravenými daty
  - Je nutno synchronizovat schéma
    - Databázové projekty ve Visual Studiu to umí
  - Občas musíme synchronizovat i data
    - Číselníky apod.
  - Testovací databáze by měla být verzována s projektem
    - Ideálně v Source Control

Typy testů (nejen integračních)

- Black box × White box
  - Black – nezajímá nás, jak to funguje uvnitř
  - White – test počítá s konkrétním algoritmem, který se používá uvnitř
- Smoke testy
  - základní sada testů ověřující základní funkce aplikace
    - Jestli aplikace vůbec nastartuje
    - Jestli nějaké okno v aplikaci nevyhazuje chybu

Testy z reálného projektu

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Testování proti reálné DB

- Typicky separátní databáze s předpřipravenými daty
  - Je nutno synchronizovat schéma
    - Databázové projekty ve Visual Studiu to umí
  - Občas musíme synchronizovat i data
    - Číselníky apod.
  - Testovací databáze by měla být verzována s projektem
    - Ideálně v Source Control

Úrovně testování

- Systémové testování
  - Aplikace se testuje jako celek
  - Ideálně
    - V reálném prostředí
    - Na reálných datech
  - Testování funkčnosti
  - Zátěžové testy
  - Testy zabezpečení

Zátěžové testy ve Visual Studiu

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Testování uživatelského rozhraní

- Od Visual Studio 2010 podpora Coded UI Tests
- Podporuje
  - Webové aplikace (libovolná technologie, prohlížeče IE a Firefox)
  - Windows Forms aplikace
  - WPF / Silverlight aplikace
  - Omezeně C++ (MFC)

Testování uživatelského rozhraní

- Nefunguje bez chyb
  - komponenty třetích stran často zlobí
- Vhodné pro jednodušší scénáře
  - Např. smoke testing webové aplikace
    - Proklikání všech stránek, jestli nějaká nepadá
    - Vhodné v kontinuální integraci
- Doinstalovat
  - Visual Studio 2010 Feature Pack 2

Coded UI testy

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Testování webových aplikací

- WatiN (http://watin.org/)
- Umožňuje z kódu
  - Otevřít prohlížeč (IE, Firefox) na určité stránce
  - Vyplňovat políčka
  - Klikat na tlačítka a odkazy
  - Kontrolovat hodnoty, vyhledávat text
- Lze užít k naskriptování složitějších scénářů

WatiN – přidání komentáře v diskusi

DEMO

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz

Testování UI

- Nepřeceňovat – neumí všechno
- Některé věci musí zkontrolovat člověk
  - Jestli se layout nerozpadá
  - Jestli jsou hodnoty na správných místech
- Nepište je hned, UI se často mění i v závěrečných fázích vývoje aplikace
  - Bude nutné měnit i testy

Jak psát testovatelný kód

- Třídy s jasně definovanými závislostmi
  - V konstruktoru
- SOLID
  - http://www.augi.cz/programovani/moderni-programovani-v-c/
- Vyhnout se statickým proměnným a metodám, kde to jen jde
  - Přípustné jsou jako syntax helpery
  - Obtížně se testují
- …

Smiřte se s tím, že

- Ne vše lze testovat
  - Ovládání různých hardwarových zařízení
- Ne vše má smysl testovat
  - Malé aplikace, kde není budoucí rozvoj
  - Často se měnící, neustálené části aplikace
- Ne vše lze testovat pohodlně
  - Složitější UI plné komponent třetích stran
    - Mnoho vztekání a ztraceného času – nevyplatí se

Doporučení

- Napsal jsem metodu, která něco počítá, a nejsem si jistý, jestli funguje
  - Místo Debug.Print a ruční kontroly napíšu test
- Pište testy alespoň pro základní a důležité aplikační bloky a třídy
- Máte chybu, která se neustále vrací? Chce to test!
- Používejte vlastní hlavu a nápady
- Nebojte se podvádět
- Zautomatizujte často opakované činnosti
- Začněte používat kontinuální integraci

Na co se ještě podívat

- Mockovací frameworky (odstřižení závislostí v unit testech)
  - Rhino Mocks nebo Moq
  - Microsoft Moles

Co byste si měli odnést

- Testování by mělo čas šetřit (v dlouhodobém horizontu)
- Nesnažte se testovat vše
  - 100% code coverage je utopie a zbytečnost
  - Testujte jen to, co má smysl
- „Jenom“ testy UI jsou lepší než nic
  - Ale měly by být doplňkem k integračním

Máte u produkčních projektů monitoring funkčnosti?

Logujete chyby vzniklé v produkčním prostředí?

A kontrolujete tyto logy pravidelně?

Diskuse

Q&A

Tomáš Herceg
Chief Software Architect @ riganti
Microsoft ASP.NET MVP
http://www.herceg.cz, http://www.vbnet.cz