Testování softwaru řízené návrhem

Matt Stephens, Doug Rosenberg (Original: *Design Driven Testing – Test Smarter, Not Harder)*

Stručně

Na jednu stranu, není kniha nijak převratná (především z testerského pohledu), na druhou stranu je to velmi zajímavý mix, ve kterém si něco zajímavého a nového najde tester, vývojář i analytik/designer. Kniha staví na velmi pragmatických přístupech k návrhu, vývoji i testování. Kdo se již vyzná v různých metodikách, může začít přílohou, což je starší přednáška jednoho z autorů. Jde o vtipnou satiru, kde si nebere žádné velké servítky s velkými, těžkými „vodopády", ale ani s extrémními agilisty. Vyplývá z toho jakási zlatá střední cesta, která je také prezentována celou toto knihou. (Přímo o metodice, na kterou se autoři odvolávají a na které staví, vyšly další publikace, které lze dohledat podle jmen autorů nebo společnosti ICONIX.)

Knihu lze nalézt na internetu i v angličtině, což může být dobrý pomocník nebo i náhrada, protože překlad především v prvních kapitolách je dost kostrbatý.

Kniha je hlavně určena vývojářům, designérům a dev-testrům. Zkušenému testerovi mohou přijít některé body samozřejmé a nezpochybnitelné. Ale zde je třeba si uvědomit, jak k testování přistupují vývojáři… ☺

Podrobněji

V první části autoři představují vzorový projekt, na kterém je jejich metodika prezentována. Pro porovnání je zde prezentována i metodika Test Driven Development - v některých bodech je popisována až příliš konfrontačně, ale většině lze dát za pravdu. Řízení návrhu pouze a jen testy a to pouze a jen unit testy má své limity, ale asi málokdo je tak ortodoxní, že by postavil seriózní vývoj pouze na TTD. (Ale jsou i tací, kteří naopak dokáží vynechat testing z životního cyklu úplně. ☺)

O co se metodika ICONIX/DDT (a tedy i tato kniha) opírá?

- Posouvá testing již do analýzy/designu aplikace (integrace v Enterprise Architect)
- Používá a mírně rozšiřuje klasický V-model. Úrovně testů jsou zde (seshora):
    - UAT
    - Testy požadavků (a to i v kódovaných testech!)
    - Testy scénářů (opět i v kódu)
    - Testy řadičů (v originále „controller", něco mezi unit testem a testem UC… řekněme logické bloky kódu, významný krok UC, metoda služby, metoda facade, …)
    - Testy jednotek – unit test (myšleny skutečné unit testy)
    - + Testy algoritmů – velmi nízkoúrovňové testy
    - *Pozn: stupeň integrace je brán jako samostatná dimenze, tzn. Test scénáře, může být jak integračním, tak zcela izolovaným testem.*
- Cílem testů je ověřit implementaci návrhu a zároveň zajištění regrese (testy jsou uvažovány jako kódované)
- Zaměřuje se spíše na pokrytí požadavků a logiky, než na pokrytí kódu (kontrast s TDD apod.)

Téměř všechny kapitoly jsou rozděleny do deseti bodů. Některé body ale souvisí s použitím Enterprise Architect a vlastního doplňku autorů do EA. Občas jsou některé skutečně do počtu, aby jich bylo vždy deset.

Například desítka osvědčených postupů – takový high-level pohled:

1. Vytvořte architekturu.
2. Dohodněte se na obchodních požadavcích a testujte vůči nim.
3. Návrh odvoďte z problémové domény.
4. Podle storyboardů uživatelského rozhraní napište případy užití.
5. Pro ověření, že případy užití pracují, napište testy scénářů.
6. Vytvořte konceptuální a podrobný návrh a testujte vůči nim.
7. Pravidelnou aktualizací udržujte model synchronizovaný s kódem.
8. Testy přijatelnosti udržujte synchronizované s kódem.
9. Mějte automatizované testy stále aktuální.
10. Kandidáta na finální verzi porovnejte s původními případy užití.

Pokud by si ne-tester odnesl z knihy alespoň tohle, super!

Testy jednotek a podrobný návrh

- Vychází především z diagramu posloupností (sequence diagram).
- Z něj se vytváří testovací případy a scénáře (jako varianty hlavního případu).
- Nevyžaduje pokrytí každého kroku, pokud je již testován jinak. (*z pohledu efektivity testování v pořádku, ale existují i argumenty proti – např. refaktorování*)
- White-box testing -> možnost stub a mock, dependency injection apod.
- Pro mimořádně náročné algoritmy nebo kriticky důležité metody psát velmi podrobné testy algoritmů – i v případě, že jsou pokryty testy na vyšší úrovni.
- Integrační testy držet v samostatné sadě, která není součástí buildu (myšleno CI, BuddyBuild, …). Nicméně sadu integračních testů udržovat aktuální a pravidelně ji spouštět.

Testování řadičů a konceptuální návrh

- Vychází z Robustness diagram (diagram robustnosti). *Nevím, jak moc je to standardní diagram, ale asi vychází z jejich metodiky. Jde o hrubý diagram chování, který se skládá z entit, procesů (controllerů, tj. řadičů), hraničních bodů a aktérů. Jde o opravdu „konceptuální“ diagram.*
- Z něj opět TC a scénáře
- Gray-box testing – neměly by být příliš závislé na implementačních detailech a tudíž odolnější při změnách detailů.
- Testy řadičů lze zřetězit. *Což má své výhody i nevýhody. Tento přístup pak může více odpovídat přístupu Behavior-Driven-Development.*
- Integrační testy držet v samostatné sadě.

Testování přijatelnosti – testy scénářů

- Vychází z Use case
- Vyžaduje, aby byly UC řádně a plně vyplněny včetně kroků a alternativní scénářů.
- Design testů a struktura testů je generován přímo z EA.
- Předpokládá se pořád ještě použití kódovaných testů!

Testování přijatelnosti – testy obchodních požadavků

- Vychází ze seznamu požadavků
- Používá doménový model
- Vytvořený seznam testovacích případů má být konzultovaný se zákazníkem!
- Uvažuje se, že i část těchto testů bude automatizovaná.
- Testy jsou zveřejňovány a sdíleny i mimo vývoj – PM, QA, zákazník, …

Další kapitoly

- Antivzory při testování jednotek a jejich řešení (kapitola 9 a 10)
    - Kapitoly především pro vývojáře a dev-testery
    - Zajímavé je, že se mezi antivzory dostal i klasický GoF vzor Singleton. *A argumentace je docela logická.*
- Automatizované integrační testování (kapitola 11)
    - Nezapomínat na testy zabezpečení
    - Výběr vhodné úrovně (V-model)
    - Psaní testů celých scénářů by nemělo brzdit projekt, pokud jsou velmi náročné.
    - Test UI zařadit mezi testy scénářů.
    - Nepodcenit náročnost tvorby a údržby
        - Parametry sítě
        - Změny struktury databáze
        - Závislost na datech (konflikty mezi testy, více testů v jedné DB, příprava a úklid, …)
        - Změny vzdálených rozhraní a chyby vzdálených systémů
    - Nepodcenit jejich hodnotu
- Testování algoritmů
- **Příloha: Alenka v říši případů užití – *vřele doporučuji*!**

Další zdroje:

- Use case driven object modelling with UML – Theory and Practice (Apress, 2007)
- Agile Development with ICONIX Process
- Extreme Programming Refactored
- JBehave, Behavior-Driven-Development
- Framework Fitnesse
- NUnitForm, Abbot, UISpec4J, FEST, Selenium
- Fuzzy Testing – Fuzzy: Brute Force Vulnerability Discovery
- Pex, Mole