Najskôr test?

PAVEL MICHLÍK

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
michlik[zavináč]gmail[.]com

Abstrakt. Správna funkčnosť je veľmi dôležitá vlastnosť kvalitného softvéru, ak nie tá najdôležitejšia. Jej zaistenie ale nie je vždy jednoduché. Je všeobecne známe, že úsilie potrebné na odstránenie chyby v softvéri je tým väčšie, čím neskôr sa chyba objaví. Vývoj riadený testami (test-driven development) je technika tvorby softvéru, ktorá bola navrhnutá s cieľom umožniť čo najrýchlejšie zistenie a opravenie chýb. Vytváraný kód sa tu testuje už od začiatku programovania, nie až po dokončení celého modulu. Tento prístup by teda mal obmedziť neskoré odstraňovanie chýb na minimum, čím sa ušetria vzácne zdroje – čas, pracovná sila aj peniaze. Samozrejme, vývoj riadený testami určite nie je univerzálnym liekom na všetko a dosiahnutie želaného efektu v žiadnom prípade nie je zaručené. V tejto eseji uvádzam výhody aj úskalia, ktoré v použití tejto techniky vidím ja. Pokúšam sa tiež odhadnúť vhodnosť jej využitia vzhľadom na druh projektu a veľkosť a organizáciu tímu vývojárov.

Úvod

Nároky dnešných zákazníkov na softvérové produkty neustále stúpajú. Vytvárané softvérové systémy sú stále rozsiahlejšie a zložitejšie a kvôli meniacemu sa prostrediu sa musia vytvárať a dodávať čo najrýchlejšie. Chyby v nich teda nie sú žiadnym prekvapením. Robí ich každý, a ak máme pred sebou softvér so státisícmi riadkov kódu, ktorý napísali nedokonalí ľudia, tak je viac ako isté, že tam nejaké chyby jednoducho musia byť. Pritom správnosť kupovaného softvéru berie zákazník ako úplnú samozrejmosť. Nebude predsa chcieť softvér, ktorý má vynikajúce používateľské rozhranie, je bezpečný, flexibilný, ale produkuje nezmysly. Na zabezpečenie tejto vlastnosti vznikla celá škála rôznych metodík testovania softvéru na viacerých úrovniach. Testovanie neraz predstavuje podstatnú časť úsilia a času, ktoré projekt spotrebuje.

V tejto práci sa venujem testovaniu softvéru na najnižšej úrovni – testovaniu súčiastok (funkčných blokov). Bežný prístup k tomuto testovaniu je taký, že najprv sa napíše kód modulu a potom sa vytvoria testy, ktorými sa funkcionalita daného modulu overuje. Existuje však iná, netradičná metóda vývoja a testovania modulov – vývoj

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-7.

riadený testami (angl. test-driven development). Očakáva sa od nej zvýšenie kvality vytváraného softvéru z viacerých hľadísk, nie len správnosti.

Vývoj riadený testami

Vývoj riadený testami je prístup k tvorbe softvéru, kedy programátor najprv vytvorí testy pre implementovaný modul a až potom píše kód samotného modulu. Počas programovania modulu potom neustále testuje, či je jeho kód správny. Takto pridávané moduly majú byť čo najmenšie a iterácie sa rýchlo opakujú.

Jedna iterácia pri vývoji riadenom testami sa skladá z nasledujúcich krokov [1]:

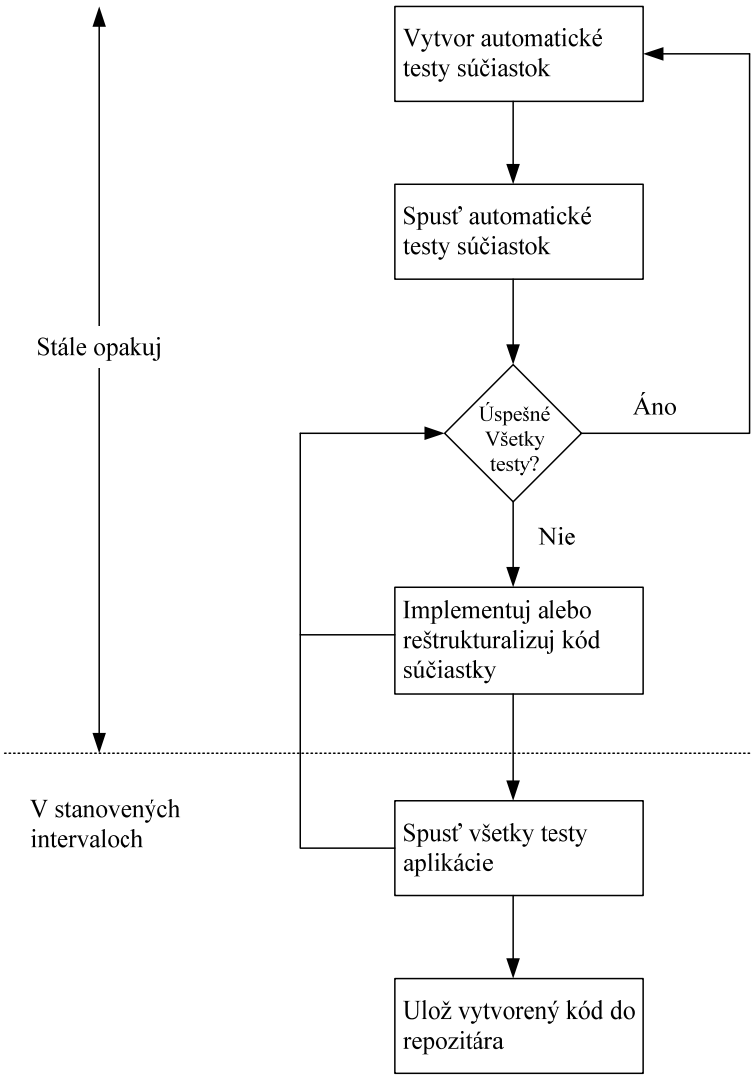
1. Vytvorenie testov pre implementovanú súčiastku. Počet týchto testov má byť len malý.
2. Spustenie testov. V tomto kroku by mali všetky testy skončiť neúspechom, keďže súčiastka ešte nie je implementovaná. Ide o jednoduchú kontrolu samotných testov – nemali by akceptovať súčiastku bez kódu ako správnu.
3. Implementovanie súčiastky. Cieľom je napísať kód, ktorý bude testami akceptovaný. V tejto fáze nie je venovaná veľká pozornosť štruktúre kódu.
4. Spustenie testov. Kroky 3 a 4 sa opakujú, až kým všetky testy pre súčiastku neskončia úspešne.
5. Reštrukturalizácia (refactoring) vytvoreného kódu. Testy pre súčiastku, vytvorené v prvom kroku, sa používajú na overenie, či nedošlo k zmene funkcionality.
6. V určených intervaloch (napríklad raz denne) sa spúšťajú testy všetkých modulov, aby sa overilo, či nové moduly nezasahujú do funkcionality iných súčastí systému.

Hlavné výhody vývoja riadeného testami sú podľa [1], [2] tieto:

- okamžitá odozva na zmeny – programátor sa ihneď dozvie, či nový kód funguje správne a či nezasiahol do funkcionality zvyšku systému
- núti k vytvoreniu podrobného návrhu až na úroveň tried a ich metód
- núti programátorov k vytváraniu takého kódu, ktorý sa dá ľahko automaticky testovať
- orientácia na úlohy – podnecuje rozkladanie práce na malé úlohy, ktoré sa dajú jednoduchšie sledovať
- testy sú stále aktuálne a sú k dispozícii pre všetky súčasti systému

Z uvedeného vidieť, že od tohto prístupu k tvorbe softvéru si vývojári sľubujú kvalitnejší produkt nielen z pohľadu správnosti, ale aj udržovateľnosti, testovateľnosti a iných dôležitých stránok kvality, ktoré sa prejavia neskôr, až bude treba vytvorený softvér meniť.

Na obr. 1 sú znázornené kroky vývoja riadeného testami.



Obr. 1. Vývoj riadený testami [1].

Experimenty a porovnanie s tradičným prístupom

Všetky vyššie spomenuté výhody vývoja riadeného testami sú zaujímavé len vtedy, ak tvorba softvéru touto metódou nevyžaduje nadmerné zdroje – pracovnú silu a čas. Experimentálne overenie je jednoduché – dva tímy programátorov dostanú tú istú

úlohu, pričom jeden tím používa bežný prístup a druhý vývoj riadený testami. Sleduje sa množstvo aj kvalita výstupov, ktoré oba tímy vytvoria. Týchto experimentov bolo vykonaných mnoho a výsledky porovnaní sa líšia. Ak sa vezmú do úvahy podmienky, v ktorých boli testy vykonané, dá sa z ich výsledkov usúdiť, kedy je nasadenie netradičného postupu výhodné a kedy nie.

Bhat a Nagappan [1] uvádzajú 2 projekty, pri ktorých bol použitý vývoj riadený testami, a ktoré boli následne porovnané s podobnými projektmi s cieľom odhadnúť vplyv tejto metódy na kvalitu vyprodukovaného softvéru a čas potrebný na vývoj. Jednalo sa o tímy 5-8 programátorov s niekoľkoročnými skúsenosťami. V oboch prípadoch bol zistený nárast kvality (kvalita bola meraná počtom chýb na 1 000 riadkov zdrojového kódu), ale aj dlhšie trvanie vývoja oproti bežnému prístupu. V prvom projekte bol zaznamenaný 2,6-krát menší výskyt chýb a odhadovaný čas vývoja bol o 25-35% dlhší. Chybovosť v druhom projekte bola nižšia dokonca 4,2-násobne a čas vývoja bol dlhší len o 15%. Pritom programátori, ktorí pracovali na prvom projekte, boli z pohľadu znalostí programovacieho jazyka aj aplikačnej domény skúsenejší. Zdá sa teda, že prínos vývoja riadeného testami je pri lepších programátoroch menší.

Vysvetlenie tohto javu je podľa mňa jednoduché. Menej skúsení programátori urobia pri rovnakej úlohe viac chýb, a teda strávia viac času ich hľadaním a opravovaním. Prístup, ktorý optimalizuje práve identifikáciu chýb, pre nich tým pádom znamená väčšiu úsporu času ako skúsenejším kolegom, ktorí robia chýb menej. Ak je program napísaný dobre hneď na prvý pokus, tak je jedno, či boli testy vytvorené pred ním alebo až po ňom, potrebný čas je rovnaký. Treba však zdôrazniť, že programátori druhého spomínaného projektu neboli v žiadnom prípade začiatočníci, mali len o niečo menej skúseností s konkrétnou doménou a programovacím jazykom. Dĺžka dovtedajšej praxe vývojárov oboch projektov bola porovnateľná.

Experiment, ktorý opísali Erdogmus, Morisio a Torchiano [2], dopadol inak. Dve skupiny študentov súčasne dostávali úlohy, pričom jedna skupina použila vývoj riadený testami a druhá skupina testovala svoj kód až po dokončení. V tomto prípade nebol zaznamenaný nárast kvality vytvoreného softvéru, a to aj napriek tomu, že pri vývoji riadenom testami programátori vytvorili viac testov. Zistilo sa ale, že vývoj riadený testami zmenšuje individuálne rozdiely v kvalite vytvoreného kódu. V skupine, ktorá vytvárala najprv program a potom testy, boli rozdiely medzi najlepšími a najslabšími programátormi oveľa viditeľnejšie. Myslím si, že táto vlastnosť môže byť užitočná pri plánovaní, keďže výsledky práce sú predvídateľnejšie. Ďalší zaujímavý záver tohto experimentu je ten, že skupina, ktorá používala vývoj riadený testami dosiahla vyššiu produktivitu. Autori to vysvetľujú lepšou dekompozíciou úlohy na časti, ktorú tento prístup vynucuje, a tým aj lepším pochopením úloh. Môže sa zdať, že tento záver je v rozpore s pozorovaním Bhata a Nagappana [1]. Ja by som tento rozdiel vysvetlil tým, že v ich prípade išlo o dobre manažovaný tím profesionálnych programátorov s niekoľkoročnými skúsenosťami v prostredí veľkej softvérovej spoločnosti. Tu pravdepodobne nedochádzalo k zlej dekompozícii problému alebo nepochopeniu úloh.

Možné problémy pri vývoji riadenom testami

Ako je zrejmé už z názvu, testy tu hrajú veľmi dôležitú úlohu a celý proces tvorby kódu je nimi ovplyvňovaný. Preto si myslím, že by sa tu mohol prejaviť základný problém všetkých testov v trochu väčšej miere ako pri iných metódach. Ide o to, že ak test skončí neúspechom, tak je niekde chyba. Ale ak test prebehne úspešne tak prichádzajú do úvahy dve možnosti: buď je program skutočne správny alebo máme zlý test. A keďže test aj samotný program vytvára ten istý človek, tak je pravdepodobné, že v obidvoch bude tá istá chyba a chybný test označí chybný program ako správny. Ak programátor zabudne overiť testom nejakú extrémnu situáciu, tak ju zabudne ošetriť aj v programe. Samozrejme, netýka sa to len vývoja riadeného testami. Tento problém sa vyskytuje vždy, keď test píše ten istý programátor, ktorý vytvoril testovaný kód. Myslím si ale, že ak si programátor overuje kód testom už počas jeho vytvárania, môže sa stať, že po prvom úspešnom ukončení všetkých testov bude mať pocit, že program je v poriadku a zabudne na ošetrenie problémových vstupov, na ktoré by mohol prísť jednoduchou prehliadkou vlastného kódu. Všetko je to samozrejme otázkou pozornosti a disciplíny, ale v časovej tiesni sa zabúda veľmi ľahko a myslím si, že zelené OK pri všetkých testoch tomu môže napomôcť. Týmto problémom sa ale nedá vyhnúť inak, ako nechať vytvorenie testov na druhého programátora. To ale vyžaduje, aby sa s úlohou podrobne oboznámili dvaja ľudia, čo by znamenalo stratu vzácneho času. A podľa môjho názoru by výsledný efekt nebol natoľko významný, aby to za ten čas stálo.

Ďalšiu možnú komplikáciu vidím v tom, že pri vývoji riadenom testami sa najprv vytvára kód, ktorý úspešne absolvuje testy, a až potom sa vykonáva reštrukturalizácia. Ak všetko prebieha podľa plánu, je to v poriadku – vytvorí sa kód, ktorý funguje správne a je aj dobre štruktúrovaný. Ak sa ale vývojári budú ponáhľať, aby stihli termíny, môžu začať vynechávať reštrukturalizáciu, aby ušetrili čas. Vznikne tak funkčný kód, ktorý zákazník prijme, a stihne sa termín. Udržovateľnosť takého kódu bude ale slabá. Je tu priestor na rozhodnutie – buď sa produkt dodá včas a reštrukturalizácia sa v prípade potreby urobí neskôr (ale už s podstatne väčšími nákladmi), alebo sa vyrobí kvalitnejší kód, ale dodá sa neskôr. Všetko závisí od toho, aký dôležitý je daný termín a aké následky vyplývajú z jeho nedodržania.

Oveľa väčšie problémy môže spôsobiť to, že sa vývoj riadený testami vnúti tímu, pre ktorý nie je vhodný. Rendell [3] vo svojom článku v časti „Pitfalls in implementing TDD“ opisuje scenár, ktorým zvyčajne končí snaha používať vývoj riadený testami, ak programátor zle pochopí jeho princípy. Zo začiatku sa vývojár začne príliš sústrediť na testy, čo mu zaberá čas, a nestíha sa dostatočne venovať samotnému kódu, ktorý má vytvoriť. Podľa Rendella sa týmto problémom nevyhnú ani tí najlepší programátori, ak s vývojom riadeným testami nemajú doteraz žiadne skúsenosti. Pri nasadzovaní preto treba počítať s tým, že zo začiatku výsledky nebudú optimálne. To ale nie je ničím zvláštnym. Prechod na nové metódy takmer vždy spôsobí dočasný pokles produktivity, kým si pracovníci ju pracovníci dokonale neosvoja. Platí to aj v iných oblastiach, nielen v softvérovom inžinierstve.

Záver

Vývoj riadený testami je určite zaujímavý spôsob, ako zaistiť kvalitu vytváraného softvéru, predovšetkým jeho správnosť, ale aj iné vlastnosti, ako napríklad testovateľnosť alebo udržovateľnosť. Nasadzovanie tejto metódy treba ale dobre premyslieť. Určite nie je vhodná pre každý projekt a za každých okolností.

Zdá sa, že vývoj riadený testami má tendenciu mierne znižovať produktivitu najlepších programátorov. Na druhej strane ale zlepšuje produktivitu tých menej skúsených. Výsledkom sú teda menšie rozdiely medzi jednotlivými programátormi, čo môže byť prínosom pre plánovanie projektov.

Hlavný problém pri nasadzovaní tejto techniky je ten, že vývojári sa ju musia najprv naučiť používať. Mnoho z nich sa zo začiatku príliš sústredí na testy, čím sa stratí veľa času. To znamená určitú počiatočnú investíciu. Produktivita a kvalita výsledkov bude optimálna až po určitom čase.

Z toho dôvodu by som pri školských tímových projektoch neodporúčal používať vývoj riadený testami, ak nemajú členovia tímu s touto technikou praktické skúsenosti. Tím by mohol doplatiť na spomínané začiatočnícke chyby, ktoré by projekt zdržali. Neexistuje tu potrebná časová rezerva a nedodržanie termínu znamená okamžitý neúspech celého projektu. Iná situácia by bola, ak by napríklad 5 členov tímu túto techniku dobre poznalo a jeden nie. Ale predpokladám, že toto nezodpovedá zloženiu bežného tímu. Takže pokiaľ oboznámenie sa s vývojom riadeným testami nie je hlavným cieľom predmetu, odporúčal by som použiť inú techniku.

Za vhodnejšie prostredie pre využite vývoja riadeného testami považujem dobre manažované tímy profesionálnych vývojárov. Myslím si, že je to priam ideálna technika pre vývoj v takých projektoch, ktoré majú vypracovanú formálnu špecifikáciu operácií. Tá je asi najlepším možným podkladom pre vytvorenie testov. Výhody vývoja riadeného testami sa tu môžu prejaviť v plnej miere.

Použitá literatúra

1. Bhat, T., Nagappan, N.: Evaluating the Efficacy of Test-Driven Development: Industrial Case Studies. In: *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, The Association for Computing Machinery, New York (2006), 356 – 363.
2. Erdogmus, H., Morisio, M., Torchiano, M.: On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, Vol. 31, No. 3 (2005) 226-237.
3. Rendell, A.: Effective and Pragmatic Test Driven Development. In: *AGILE Conference* (2008), 298-303.

Annotation

Test first?

Correctness is a very important characteristic of a quality software application, if not the most important one. But ensuring correctness can be a difficult task. It is a known fact that the effort that is needed repair a defect in software is bigger, when the defect is discovered late. Test-driven development is a software development technique, which was designed to allow early defect discovery and repair. The code is tested by the programmer from the very beginning, unlike traditional techniques, where the code is tested after the whole module completed. Late defect repairs should be minimized this way. That means saving expensive resources – time, people and money. Of course, test-driven development is not a universal solution to everything and the desired effect is not guaranteed at all. In this essay I present my personal view on its advantages and disadvantages. I also attempt to describe situations where test-driven development helps creating better software and where not.