TESTAMI RIADENÝ VÝVOJ: EFEKTÍVNY SPÔSOB PROGRAMOVANIA?

„Najprv testy, potom kód!“ – K. Beck

Matej Maruš

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

Abstrakt. *Testovanie softvéru sa nemusí vykonať až po naprogramovaní zdrojového kódu. S príchodom agilného spôsobu vývoja softvéru a extrémneho programovania sa testovanie presunulo z posledných fáz vývoja softvéru na popredné fázy. Testovanie softvéru dostalo ďalšie kompetencie v podobe určovania vlastností kódu a smerovania vývoja malých časti kódu. Hlavnú otázku, ktorú sa snažím v eseji zodpovedať je, do akej miery je vhodné použiť testami riadený vývoj v tímovom projekte. Veľkú časť vedomostí čerpám z prieskumov, ktoré boli zrealizované na amerických renomovaných univerzitách a tým sa aspoň z časti približujem k podmienkam, ktoré máme my na univerzite. Teoretická časť eseje sa zameriava na popis fungovania testami riadeného vývoja.*

Kľúčové slová: *testami riadený vývoj, testy, kvalita, prieskum, extrémne programovanie, agilné metódy vývoja.*

Úvod

Na začiatku som poriadne nevedel presne definovať význam slova test, aj keď je to triviálne slovo, ktoré sa používa v každodennej reči. Po vyhľadaní definície v krátkom slovníku slovenského jazyka som bol trochu múdrejší. Definícia znie *„skúška na zistenie schopností, vedomostí, hodnôt alebo vlastností“*[5]. Zaujala ma hlavne časť, ktorá hovorí o skúške na zistenie hodnôt alebo vlastností. Ďalej mi presne nebolo jasné, čo znamená slovo kvalita, tak isto som sa pozrel do slovníka a našiel som toto: *„súhrn vlastností*

odlišujúcich predmet, jav od iných"[5]. Ak spojíme tieto dve slová dokopy, test kvality, dostaneme skúšku vlastností, ktoré odlišujú predmet od iných. Vyzbrojený touto vedomosťou sa môžeme pustiť do písania eseje zaoberajúcej sa testovaním kvality.

Určite je každému jasné, že testy sú neoddeliteľnou súčasťou ľudského pokroku a vývoja už dlhú dobu. Trúfol by som si povedať, že v každom priemyselnom odvetví, zahŕňa proces výroby v nejakej forme testy kvality, pomocou ktorých môžeme overiť, či sa nám podarilo vyhotoviť funkčný produkt, ktorý spĺňa všetky nami zadané kritéria. Osobitnou kategóriou je zdravotníctvo, v ktorom dôležitú úlohu hrá hlavne overovanie vhodnosti použitého lieku alebo liečebnej metódy pomocou testov. Absencia testov by znamenala vážne poškodenie zdravia pacienta, ba i ohrozenie jeho života. A ľudský život je to najcennejšie, čo môžeme mať. Preto sa už veľmi dávno zaviedli testy, pomocou ktorých si môžeme overiť kvalitu (v zdravotníctve skôr overiť vhodnosť použitia metódy, či lieku), čiže do akej miery spĺňa medikament podmienky, ktoré sme si definovali na začiatku. Testy sú preto veľmi dôležité a nenahraditeľné.

Ďalšie dôležité slovo, ktoré sa bude často skloňovať v tejto práci je kvalita. Tento pojem na nás útočí z každej strany. Niekedy mám pocit, že celá naša spoločnosť je kvalitou doslova posadnutá. Ale je kvalita, až taká dôležitá? Veď podstata kvality spočíva v tom, čo si definujeme ako cieľ. Ak si definujeme podmienku, že softvér má obsahovať veľa chýb, teoreticky dospejeme k záveru, že čím bude horší, tým bude kvalitnejší. Ľudia by si mali vyberať produkty, nie na základe tvrdenia reklamy o zaručenej kvalite, ale skôr by sa mali zamyslieť nad podmienkami, ktoré určujú kvalitu pre nich podstatnú.

V tejto eseji sa chcem venovať oblasti zaoberajúcej sa testami riadeného vývoja softvéru. V prvej časti eseje sa budem snažiť vysvetliť, čo znamená testami riadený vývoj a opíšem základný spôsob fungovania pomocou obrázku. V ďalšej časti eseje sa venujem výhodám a nevýhodám, ktoré prináša testami riadený vývoj. Poznatky sú čerpané z rôznych výskumov, ktoré boli uskutočnené na renomovaných univerzitách. Tieto výsledky získané z výskumov sa snažím zhrnúť v tabuľke. V ďalšej časti sa zamýšľam aj nad rôznymi zaujímavými výsledkami prieskumov, ako napríklad vplyv testami riadeného vývoja na celkovú cenu projektu alebo obľúbenosť tohto štýlu tvorby softvéru u menej skúsených programátorov. V závere sa snažím zamyslieť sa nad vhodnosťou použitia testami riadeného vývoja v tímovom projekte.

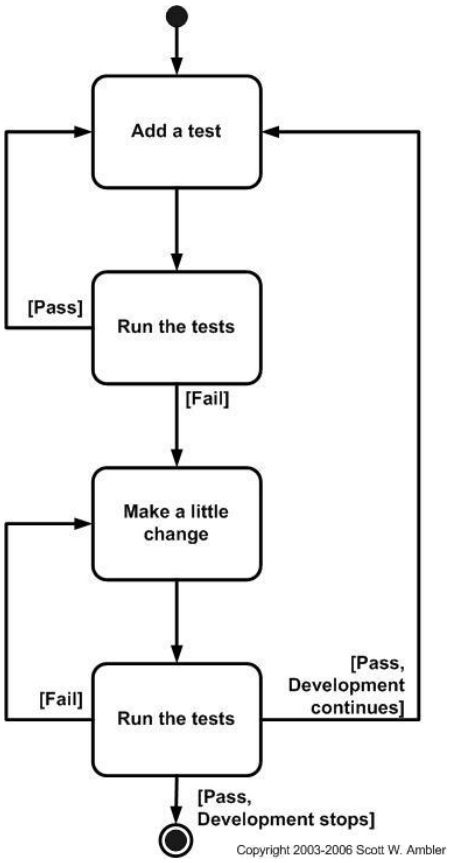
Nekonečný príbeh: červená, programuj, zelená, úprava

Myslím si, že agilný spôsob vývoja softvéru v ostatnej dobe naberá na popularite hlavne v malých tímoch venujúcich sa menej rozsiahlym projektom. Testami riadený vývoj patrí do agilného spôsobu vývoja a je jednou z dvanástich kľúčových praktík extrémneho programovania[3].

Vývoj riadený testami (angl. Test driven development) priniesol novú stratégiu overovania kódu ešte pred samotným napísaním, čo by len kúsku kódu. Testami riadený vývoj je známy aj pod inými názvami ako *test-first programing* alebo *test-first design*. Aj keď nám to názov naznačuje, nejedná sa iba o testovaciu techniku. Zameriava sa na celý proces vývoja softvéru. Do značnej miery by sa dalo povedať, že je to návrhová technika, ktorá je súčasťou celého procesu vývoja softvéru[1]. Návrh testov je prvý krok v rozhodovaní, čo má program nevyhnutne vykonávať. Preto sa táto časť zaradzuje aj do analýzy vývoja

softvéru. V tejto analýze získavame požiadavky aj od zákazníka, a tým pádom zapájame do testami riadeného vývoja aj externé osoby, ktoré sa nezúčastňujú konkrétneho písania programovacieho kódu[3].

Vývoj riadený testami vo svojej podstate núti vývojárov vytvárať a používať automatické testy. Keďže celá logika je postavaná na opakovanom testovaní malých kúskov kódu, ktoré píšeme v krátkych iteráciách. Ručným testovaním by bolo veľmi náročné otestovať každú novú funkcionalitu a hlavne by to bol časovo nedosiahnuteľný cieľ. Na obr. 1. je znázornená postupnosť krokov, ktoré sa vykonávajú v každej iterácii testami riadeného vývoja.



Obr. 1. Proces testovania riadeného vývoja.

Postup jednotlivých fáz testami riadeného vývoja sa dá zhrnúť do šiestich bodov:

1. Pridaj nový test (Add a test)
2. Spusti všetky testy a zisti či nový zlyhal (Run the tests)
3. Implementuj novú funkcionalitu (Make a little change)
4. Spusti testy a zisti, či sú všetky úspešné (Run the tests)
5. Uprav celkový kód (Refactoring)
6. Opakuj (Development continues)

V prvej fáze by mal test overovať novú funkcionalitu, ktorú pridávame do existujúceho kódu. Pridať môžeme v jednej iterácií aj viac testov. Jedna iterácia by nemala obsahovať veľké zmeny v programovacom kóde a mala by byť zameraná iba na implementáciu jednej novej funkcionality. Požiadavky sa získavajú z prípadov použitia, ktoré sme si zadefinovali spolu so zákazníkom.

V druhej fáze musí nový test zlyhať. Ak by bol test splnený, chyba nastala v definovaní testu a treba vykonať nápravu redefináciou nového testu. Táto fáza je príznačná červenou farbou pri spustení testu.

V tretej fáze sa vytvára nová časť programovacieho kódu, ktorá by naplnila požiadavky pri zadefinovaní nového testu. Myslím si, že táto fáza je podstatná z viacerých pohľadov. Prvý pohľad je, že v tejto fáze sa vytvára konkrétny kód, ktorý chceme naprogramovať. Testy sú len podporné prvky na dosiahnutie cieľa t.j. vytvorenie softvéru, ktorý si zákazník objednal.

Štvrtá fáza je špecifická zelenou farbou pre všetky testy, ktoré sú definovane od začiatku projektu. Zelená znamená, že všetky testy sú úspešné a programovací kód spĺňa všetky požiadavky.

V piatej fáze nastáva upratovanie v kóde. Vykonaním potrebných zmien sa sprehľadní celý systém. Upratovaním (angl. refaktoring) v každej iterácii dostaneme programovací kód do dobrej konzistencie. Nachádzajú sa návrhové vzory a odstraňujú sa pachy (angl. bad smells).

Posledná fáza sa zameriava na opakovanie celého procesu. Konkrétna iterácia je na konci, a môžu sa pridávať ďalšie funkcionality. Alebo je celý proces ukončený a odovzdaný do používania, respektíve odoslaný na ďalšie doplňujúce testy.

Rôzni autori implementujú niektoré časti procesu ináč. Najviditeľnejšie rozdiely sú vo fáze štyri. V článku[1], ktorý sa venuje testami riadeného vývoja je táto metóda pozmenená v tom, že po implementácii kódu, sa nespúšťajú všetky testy, ale iba nový test. Nezávisle od jednotlivých programátorov sú testy spúšťané v pravidelných intervaloch. Odporúčané je minimálne aspoň raz denne, aby sme overili konzistenciu kódu ako celku. Otázka znie: je táto modifikácia opodstatnená? Myslím si, že áno. Pri veľmi veľkých projektoch je limitujúci čas vykonávania testov. Ak by len jeden test trval päť sekúnd a takýchto testov by bolo tisíc, každé otestovanie by zabralo cca osem minút. To je neprípustný čas, pri zohľadnení priemernej ceny práce programátora. V nasledujúcej tabuľke opisujem tri prieskumy, ktoré boli vykonané na univerzite.

Tab. 1. Prieskum používania testami riadeného vývoja

Stupeň zručnosti študenta	Počet študentov	Produktivita študentov	Kvalita programov
začiatočník	24	52% nárast	Žiaden efekt
pokročilí	18	24,5 nárast	34,8% menej chýb
pokročilí	240	78% nárast	67% zlepšenie kódu

Vedomosti z týchto prieskumov som zhrnul vo výhodách a nevýhodách testami riadeného vývoja.

Výhody testami riadeného vývoja

- Programátori sú nútení zamyslieť sa nad funkcionalitou kódu, predtým ako sú ovplyvnení spôsobom naprogramovania. Inými slovami programátori nie sú obmedzovaní svojimi znalosťami a sú schopní navrhnúť logiku programovacieho kódu lepšie.
- Zvýšená kvalita kódu: je zabezpečovaná neustálym testovaním celého systému. Počtom testov sa kvalita softvéru zlepšuje, odhliadnuc od časovej náročnosti testovania. Zvýšená kvalita je dosiahnutá aj písaním testov pred samotnou implementáciou.
- Redukcia chýb: podľa štúdii zameraných na odhad chýb v projektoch sa v testami riadeného vývoja nachádza od 35% do 45% menej chýb ako v iných projektoch využívajúcich odlišné testovacie techniky[1]. V iných štúdiách je odhad zmenšenia počtu chýb použitím testami riadeného vývoja od 18% do 50%[4].
- Zlepšená produktivita: niektoré prieskumy dokazujú obrovské zvýšenie produktivity od 25% až do 50%[1]. Tejto problematike sa venujem v ďalšej časti eseje.
- Lepšie porozumenie kódu a dôvera vo vykonaných zmenách: prieskum medzi študentmi dokazuje, že použitím testami riadeného vývoja dochádza k vyššiemu porozumeniu implementovaného kódu a programátori majú menej strachu pri zmene niektorej časti kódu. Táto výhoda bola markantnejšia viac u starších a skúsenejších programátorov ako mladších a neskúsených.
- Pozitívne vplýva na psychiku: postupné zvyšovanie množstva splnených testov evokujú u programátora progres pri implementácii a tak programátor neupadá do depresívnych stavov zapríčinených stagnáciou.
- Sústredenie na splnenie testu: programátor venuje celé svoje úsilie k splneniu testu a nezaoberá sa inými vlastnosťami softvéru. Inými slovami, nesnaží sa míňať čas na implementovanie rôznych nových vlastností, ktoré nemá definované, ale zameriava sa výhradne na splnenie testu.
- Záznam procesu implementácie: testy ponúkajú informácie o projekte. Zaznamenávajú, ktoré vlastnosti boli kedy vytvorené a vytvárajú dokumentáciu k celému projektu.

Nevýhody testami riadeného vývoja

- Testami riadený vývoj je nevhodný pre začínajúcich programátorov. Naučenie sa testovania zaberá čas a vyžaduje rozsiahlejšie skúsenosti z programovania.
- Zníženie produktivity. Výsledky niektorých štúdií dokazujú zníženie produktivity o 5% až 10% zapríčinenej časom potrebným k písaniu testov[1].
- Programátor musí vidieť pridanú hodnotu v nefunkcionálnom programovaní (testovací kód). Nie všetci programátori sú schopní si osvojiť techniku písania testov pred samotným napísaním kódu.

Je vhodne použiť testami riadený vývoj pre tímový projekt?

Na túto otázku odpoviem až v závere tejto eseje, keďže si myslím, že nemáte dostatok informácií na správne rozhodnutie sa. Najprv chcem ešte rozobrať vyššie spomínané prieskumy. Získané znalosti môžu dopomôcť tímu programátorov k lepšej a hladšej aplikácii testami riadeného vývoja. Tým sa vyhnú problémom, ktoré sa objavujú medzi inými študentmi začínajúcimi s touto metodikou vývoja softvéru. Prieskumy boli vykonávané medzi študentmi, ktorí mali rôzne najvyššie dosiahnuté vzdelanie a prekvapujúco táto rozdielnosť do značnej miery vplývala na efektivitu a kvalitu výsledného kódu. Starší a skúsenejší študenti si skôr osvojili vyvíjať softvér pomocou testami riadeného vývoja.

Po naštudovaní tejto problematiky ma napadajú rôzne otázky. Je potrebne ešte použiť nejaké iné metódy ako testami riadený vývoj? Na takto položenu otázku sa dá odpovedať áno, aj nie. Myslím si, že malé projekty (polročná práca na projekte s maximálne siedmimi členmi) nepotrebujú iné testovanie a vystačia si s automatickými testami, ktoré kvalitne otestujú všetky funkcie. Veľké projekty, na ktorých pracuje množstvo tímov potrebujú komplexnejšie testy, ktoré otestujú celkovú funkcionalitu , výkonnosť systému, systémovú integráciu a taktiež by mali otestovať zmeny validované pomocou akceptačných testov.

Je potrebná dokumentácia? V kvalitných projektoch je nevyhnutné vytvoriť dokumentáciu k vyvíjanému softvéru. Svojim spôsobom testami riadený vývoj obsahuje dokumentáciu, ktorú je možné nájsť priamo v testoch. K veľkému projektu je potrebné vyhotoviť aj ďalšie zložky dokumentácie, ako napríklad návrh celkového systému a iné.

Rád by som Vás ešte oboznámil s jedným prieskumom, ktorý sa v tabuľke nenachádza. Tento prieskum bol realizovaný na vzorke sto študentov začiatočníkov, a zameraný na obľúbenosť testami riadeného vývoja. Študenti dostali možnosť zvoliť si, či budú testovať programovací kód po implementácii alebo napíšu test pred implementáciou. Iba 10% študentov si zvolilo písanie testu pred samotnou implementáciou[1]. Je testami riadený vývoj neobľúbený? Z vlastnej skúsenosti viem, že je veľmi ťažké najprv si vytvoriť test, a preto ma vôbec neprekvapuje výsledok prieskumu. Sám by som sa zaradil medzi programátorov, ktorí ešte nepoužívajú všetky praktiky softvérového inžinierstva, a preto je náročné si osvojiť agilný spôsob vývoja.

Z prieskumov vyplynulo, že najobľúbenejší programovací jazyk je Java a študenti vytvárali testy pomocou jUnits, ktorý je vhodný na automatické testovanie malých časti kódu v procese vývoja softvéru pomocou testov.

Počas písania odbornej eseje som sledoval, či je testami riadený vývoj používaný v našom tímovom projekte. Po ukončení prvého šprintu môžem povedať, že nikto z nás nevytvoril testy pred začatím implementácie. Nie je to nič prekvapujúce, keďže ako z prieskumov vyplýva, študenti začiatočníci nemajú v obľube agilný spôsob vývoja, ak nemajú ešte dostatok skúsenosti v programovaní, a nemajú zaužívané praktiky softvérového inžinierstva. Myslím si, že správny čas na nasadenie extrémneho programovania nastáva až po piatich rokoch, keď študent začína aktívne programovať a získava zručnosti na určitej úrovni. Pre úplnosť musím dodať, že v tíme sa nachádza sedem študentov a testami riadený vývoj bol prikázaný ako stratégia vývoja, ktorú majú využívať pri vytváraní projektu.

Každého správneho manažéra, ktorý sa riadi princípmi trhového hospodárstva a chce minimalizovať náklady a maximalizovať zisky musí napadnúť otázka ohľadne peňazí. Prispieva teda testami riadený vývoj k ušetreniu finančných prostriedkov? Odpoveď sa nachádza v prieskumoch. Aj keď nie je explicitne povedané, či sa ušetria peniaze, z výsledok prieskumov sa to dá odhadnúť. Ak sa zvýši produktivita a zníži sa počet možných chýb v produkte, má to za následok zníženie celkovej ceny nákladov(menej času potrebného na vývoj). V konečnom dôsledku dochádza k šetreniu, a tým aj k navýšeniu zisku, pod podmienkou využitia skúsených programátorov, ktorí sú oboznámení a aktívne používajú agilný spôsob vývoja na základe testami riadeného vývoja.

Záver

Má zmysel používať testami riadený vývoj v tímovom projekte? Bolo by veľmi alibistické odpovedať nie a tak zahodiť kvalitnú stratégiu zameranú na vývoj softvéru. Myslím si, že tímový projekt má za cieľ naučiť študentov používať inžinierske postupy a zaužívať si ich. Preto si myslím, že je ten najvhodnejší čas. Aj keď to bude z počiatku ťažké, netreba sa báť a smelo sa pustiť do vývoja riadeného testami.

Získané informácie v tejto eseji budú určite využiteľné v tímovom projekte. Na základe prieskumov sa budem snažiť, aby sme si čím skôr osvojili túto praktiku vývoja softvéru a tak dosiahli vyššiu kvalitu a menej chýb. V neposlednom rade programátor získava väčšiu dôveru vo svoj kód. To by sám rád dosiahol aj ja. Budem sa snažiť presviedčať kolegov o vhodnosti použitia testami riadeného vývoja na menšie projekty. Tímový projekt by som zaradil medzi tie menej rozsiahle, ktoré sú určené skôr ako projekty, na ktorých si môžeme vyskúšať všetky praktiky spojené s vývojom softvéru a tak sa učiť za behu.

Použitá literatúra

1. Bhat, T., Nagappan, N.: Evaluating the Efficacy of Test-Driven Development: Industrial Case Studies. Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, September 2006, s. 356-363.
2. Desai, Ch., Janzen, D., Savage, K.: A Survey of Evidence for TestDrivenDevelopment in Academia. ACM SIGCSE Bulletin, vol. 40 no. 2, June 2008, s. 97-101.
3. Janzen, D., Saiedian, H.: Test-Driven Development: Concepts, Taxonomy, and Future Direction. Computer, vol. 38 no. 9, September 2005, s. 43-50.
4. Jones, Ch., G.: Test-driven development goes to school. Journal of Computing Sciences in Colleges, vol.20 no.1, October 2004, s. 220-231.
5. Kačala, J., Pisarčíková, M., Považaj, M. et al.: Krátky slovník slovenského jazyka. 4. rev. vyd., VEDA, 2003. s. 944.

Annotation

Test driven development: effective way to develop programming code?

Software testing is not performed until source code has been programmed. With the rise of agile software development methods called Extreme Programming testing is moved from the last phases of software development to the "heart" of development. Software testing has received additional responsibilities in terms of determining the characteristics of the source code and directions of development of small pieces of code. The main question that I try to answer the essay is the extent to which it is appropriate to use the test-driven development in a team project. Large part of knowledge comes from surveys which were implemented at American universities and thus at least partially meet the conditions that we have at our university. The theoretical part of this essay focuses on the description of the tests driven development method.