O krok pred chybami

JAKUB KOŽÍŠEK

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jakub.kozisek@gmail.com

Abstrakt. Súčasná doba je charakterizovaná požiadavkami na rýchlu a lacnú tvorbu vysoko kvalitných aplikácií. Tomuto trendu sa snažia prispôsobiť aj samotné procesy vývoja softvéru, ktoré definujú jednotlivé fázy tak, aby sa do maximálnej možnej miery zabránilo potencionálnym problémom, ktoré by mohli vývoj spomaliť. Zlé pochopenie požiadaviek zákazníka, ich nestálosť ako aj chybovosť produktu - to všetko sú príčiny zdržaní, ako aj nárastu nákladov. Vývoj riadený testovaním ("Test-driven development", TDD) v kombinácii s akceptačnými testami je technika, ktorá sa snaží týmto problémom predchádzať a nachádza stále viac prívržencov v oblasti vývoja softvéru. Esej poukazuje na problémy klasických prístupov k testovaniu a problémy s komunikáciou medzi zákazníkom a vývojovým tímom, približuje vývoj riadený testovaním, snaží sa demonštrovať jeho výhody zhodnotením výsledkov viacerých štúdií a zároveň predstavuje vývoj riadený testovaním i z pohľadu, ktorý nemusí byť ihneď zrejmý – ako nástroj komunikácie a motivácie.

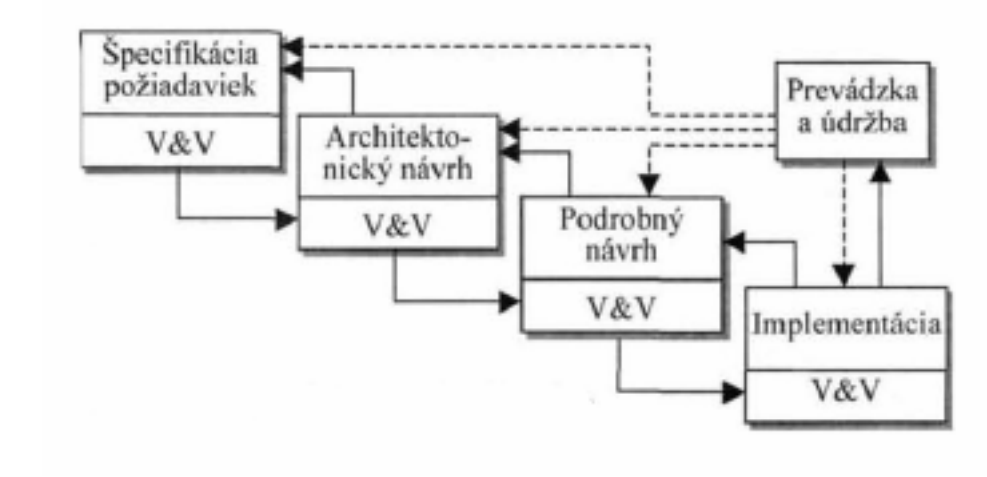
Úvod

Súčasnosť je v znamení techniky, a teda i softvérových produktov. Existuje mnoho projektov, ktoré sa snažia prísť na dynamicky sa vyvíjajúci trh s novým riešením ako prvé, a tým si získať čo najväčší počet klientov. Je zrejmé, že v dnešnom vysoko-konkurenčnom prostredí patrí medzi najdôležitejšie faktory kvalita produktu. Za účelom jej zabezpečenia sa stalo štandardom testovanie, ktoré vo väčšine prípadov nasleduje za implementáciou. Tento spôsob testovania so sebou prináša úskalia, a preto po čase vznikol alternatívny prístup, ktorý sa snaží predchádzať mnohým problémom a zároveň získava dodatočné výhody, ktoré vyplývajú z jeho pozície v samotnom procese.

Manažment v softvérovom inžinierstve, október 2007, s. 1-1.

Klasické problémy testovania a komunikácie

Jedným z najrozšírenejších modelov vývoja softvéru je vodopádový model (pozri Obr. 1). V situácii, kedy sú všetky požiadavky zrejmé už na začiatku a s časom sa nemenia, je tento prístup vysoko efektívny. V reálnych podmienkach je však bežné, že samotný zákazník nie je schopný na začiatku vývoja určiť presne, čo požaduje. On sám postupne zdokonaľuje svoju vlastnú predstavu o riešení. Vodopádový model nie je natoľko flexibilný, aby umožňoval požiadavky postupne dopĺňať a aktualizovať. Štúdie ukázali, že typická špecifikácia požiadaviek je 15% kompletná, 7% korektná a jej oprava a kompletizácia je cenovo neefektívna [2].



Obr. 1 Vodopádový model [1]

Zaužívanou pozíciou testovania v procese vývoja softvéru je jeho nasledovanie za implementáciou. Úspešná realizácia testovania vedie k nasadeniu projektu, ktorý býva do konca svojho životného cyklu udržiavaný. Tento fakt prináša so sebou dve skupiny problémov, a to:

1. psychologické
2. komunikačné

Implementácia z pohľadu vývojára je sťažená faktom, že nemá možnosť pravidelne kontrolovať správanie súčiastok, ktoré vyvíja. Keď nastanú situácie vyžadujúce si zmeny v zdrojovom kóde s účelom jeho sprehľadnenia alebo zefektívnenia (tzv. „refaktoring“), môže tento fakt vystupovať ako zábrana. Bez pravidelnej kontroly sa stáva, že s narastajúcou zložitosťou autor postupne stráca

dôveru vo svoj vlastný výtvor a čoraz menej rád zdrojový kód rozširuje a mení. Tieto problémy sú odpozorované z vlastnej skúsenosti. Sám som bol v minulosti zodpovedný za vývoj súčiastok, na ktoré ešte testy neexistovali, no boli na nich požadované rozsiahle zmeny.

Ďalším psychologickým problémom v situáciách, keď skupina ľudí zodpovedných za vývoj je taktiež zodpovedná za testovanie, je fakt, že si môžu často aj podvedome voliť cestu „ľahšieho odporu“. Sústreďujú sa na menej náročné časti súčiastok a zložité neprehľadné konštrukcie pri testovaní nepokrývajú dostatočne, čo môže viesť v budúcnosti k nákladným opravám, pretože chyby sú odhalené až v neskorých fázach životného cyklu projektu.

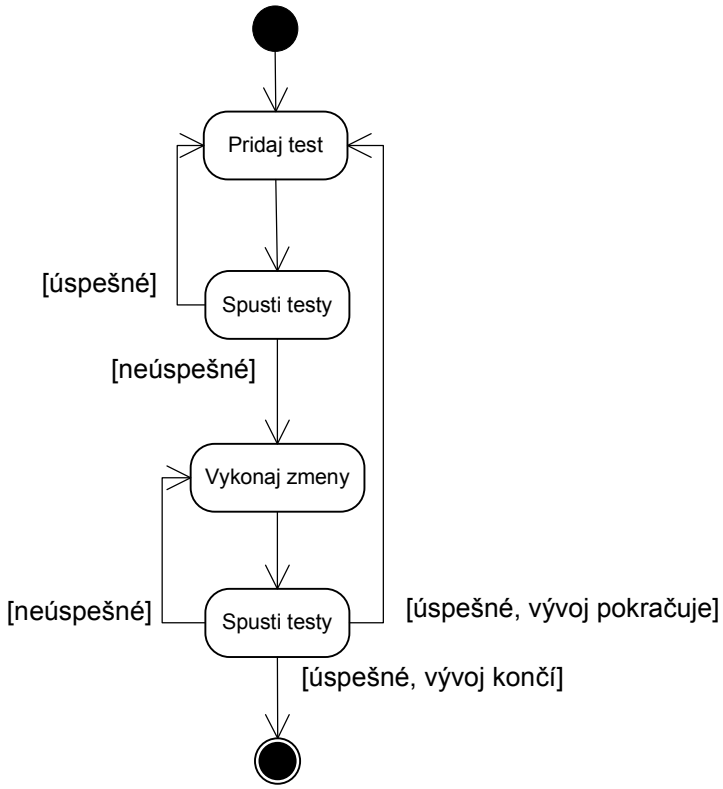
V modeloch vyžadujúcich rozsiahlu špecifikáciu požiadaviek sa nekladie dôraz na častú komunikáciu medzi zákazníkom a vývojovým tímom. Špecifikácia požiadaviek by mala zabezpečiť jasný a kompletný obraz očakávaní zákazníka. Už vyššie bolo spomenuté, že to tak vo významnej väčšine prípadov nie je. Testovanie je v modeloch závislých od nemennej špecifikácie často miestom, kedy má zákazník možnosť prvýkrát sledovať realizáciu jeho požiadaviek. Čo ak špecifikované požiadavky a jeho očakávania sú nakoniec rozdielne alebo boli nesprávne pochopené? Systém nepracuje tak, ako by mal, teda sa jedná o chybu. Oproti „klasickým“ chybám, ktoré vychádzajú z nekorektnej implementácie je táto už v samotnej špecifikácii a vyžaduje si ešte väčšie úsilie na opravu, pretože môžu byť potrebné aj zmeny návrhu a celkový dopad na rozpočet aj čas je teda mnohonásobne vyšší.

Vývoj riadený testovaním a akceptačné testy

Vývoj riadený testovaním (TDD) je technika, ktorá pristupuje k testovaniu diametrálne odlišne – umiestňuje ho totiž už pred implementáciu. Testovanie pri tomto prístupe sa delí na tzv. testy jednotiek (“unit” testy) a akceptačné testy. Testy jednotiek informujú, či jednotlivé súčiastky pracujú správne, teda slúžia na kontrolu, či implementácia vyhovuje návrhu. Akceptačné testy sa nezameriavajú na systém po častiach, ale skúmajú ho ako jeden celok. Na základe požiadaviek zákazníka sa spolu s ním pripravia scenáre, ktoré dané požiadavky otestujú a scenáre sa implementujú do formy testov. Ak sa teda úplná množina akceptačných testov vykoná úspešne, boli všetky požiadavky zákazníka na projekt splnené a vývoj sa dá považovať za ukončený. Samozrejme na to, aby to platilo, je potrebné, aby na každú požiadavku existoval scenár, ktorý preskúma jej naplnenie. I tu sa stáva, že scenáre nie sú úplné, avšak na rozdiel od mnohých iných prístupov je komunikácia so zákazníkom častá a pravidelná, a preto sa minimalizuje výskyt takejto situácie.

Princíp práce pri vývoji riadenom testovaním je znázornený na Obr. 2. Pred samotnou tvorbou implementácie je vytvorená počiatočná množina testov. Keďže testy si vyžadujú implementáciu, zo začiatku ich nie je možné ani preložiť do spustiteľnej formy, a teda ich spustenie je neúspešné. Testovaná funkcionalita je následne implementovaná a testy sú spustené znova. Tieto kroky sa vykonávajú, pokiaľ testy neprebehnú úspešne. V tom momente sa pridajú nové testy a celý kolobeh sa opakuje, až kým nie je vývoj ukončený. V prípade, že pre pridané testy existuje patričná

korektná funkcionalita, pridávajú sa testy ďalšie [3]. Tento prístup umožňuje rast implementácie po častiach, pričom je zaručené, že každý prírastok je už otestovaný.



Obr. 2 Kroky pri vývoji riadenom testovaním

Vývoj riadený testovaním prináša novú dimenziu výhod [4]:

1. testy slúžia ako vykonateľná špecifikácia - požiadavky zákazníka sú „pretavené“ do spustiteľnej formy a ich plnenie môže byť pravidelne a automaticky overované,
2. testy jednotiek dodávajú dôveru vývojárom vo svoj kód, ktorý rastie postupne a jeho správnosť je sledovateľná; akceptačné testy ukazujú zákazníkovi a vedeniu, že vývoj napreduje správnym smerom a že ho môžu jednoducho sledovať, taktiež ho podporujú vo vkladaní časových a finančných prostriedkov do projektu [5],
3. vývojári sa postupne učia, ako systém stavať na základe daných požiadaviek a zákazník má možnosť postupne zisťovať, čo požaduje,

4. keďže rozhrania implementovaných súčiastok a ich použitie sú definované v testoch a na ich základe dochádza k vývoju, je možné považovať testy za dokumentáciu správania implementácie a zároveň príklad jej využitia,
5. nedochádza k prebytočnej implementácii, t.j. nie je implementovaná taká funkcionalita, ktorá by nebola požadovaná testami.

Z vyššie uvedených bodov je zrejmé, že testy dostávajú charakter komunikačného i dokumentačného nástroja. Zároveň je možné ich pokladať za zdroj motivácie v pokračovaní vývoja, pretože tento prístup eliminuje psychologické problémy spomenuté v predchádzajúcej časti. Rozsiahle pokrytie testami nielen udáva vývoju jasný smer, ale robí proces refaktoringu bezpečnejším, pretože je možné čiastkovo sledovať, či nedošlo k neželanej zmene správania. Táto technika teda prispieva k zvýšeniu kvality implementácie z dlhodobého hľadiska.

Experimenty

S rastúcim záujmom o vývoj riadený testovaním a jeho uplatnením v praxi začali vznikať aj experimenty, ktoré sa snažili zmerať prínosy tejto techniky z časového hľadiska i z hľadiska celkového dopadu na kvalitu softvérového produktu.

Müller a Hagner vykonali porovnanie tradičného prístupu a prispôsobeného vývoja riadeného testovaním na vzorke deväťnástich absolventov [6], pričom merali efektivitu z troch hľadísk:

1. času potrebného na vývoj
2. výslednej kvality zdrojového kódu
3. zrozumiteľnosti

Absolventi boli rozdelení na dve skupiny. Každá zo skupín riešila rovnakú úlohu, pričom im bola poskytnutá špecifikácia i dizajn. Vývoj pozostával z dvoch fáz: z implementačnej a akceptačnej. Jedna skupina implementovala systém zároveň s tvorbou testov. Druhá dokončila najprv implementáciu a potom vytvárala testy. Po vyhodnotení úspešných akceptačných testov sa zistilo, že čas na vývoj bol u oboch tímov približne rovnaký, kvalita kódu po oboch fázach taktiež, avšak skupina využívajúca vývoj riadený testovaním mala významne menej chýb v kóde pri znovuvyužití kódu. Tvorbou testov a následnou implementáciou sa zrozumiteľnosť kódu zvýšila, čo viedlo ku korektnému využitiu existujúcich rozhraní.

Výsledky sa môžu zdať v určitých smeroch prekvapujúce, avšak treba brať do úvahy, že študenti mali obmedzené skúsenosti s vývojom riadeným testovaním a vzorka bola malá.

Druhý experiment bol vykonaný pánom Georgom [7], ktorý využil profesionálnych vývojárov, aby zistil, do akej miery majú skúsenosti dopad na jednotlivé merané veličiny. Využil 8-členné vývojové tímy a rozdelil ich na dvojice. Polovica dvojíc využívala konvenčné postupy (dizajn, vývoj, testovanie, ladenie) a zvyšné dvojice použili techniku vývoja riadeného testovaním. Dvojice pracovali

štýlom „programovanie v páre“, kedy dvaja vývojári sedia za jedným počítačom a spolupracujú.

Na testovanie výstupu dvojíc bolo použitých dvadsať testov metódou „čiernej skrinky“, pričom sa zisťovala miera naplnenia požiadaviek i ošetrenie chýb. Namerané hodnoty ukázali, že dvojice využívajúce moderný prístup prešli priemerne 18% viac testami. Meranie produktivity skončilo v prospech tímov vyvíjajúcich testy až po implementácii (TDD tímy pracovali v priemere o 19% dlhšie), avšak z nich iba jeden tým poskytol testy dostatočne pokrývajúce implementovanú funkcionalitu.

Jasná výhoda vývoja riadeného testovaním vyplývala z poslednej meranej veličiny – z pokrytia kódu testami (“code coverage”). Kým za priemyselný štandard pokrytia sa považujú hodnoty v rozpätí 80% až 90%, tímy implementujúce na základe testov dosiahli v priemere 98% pokrytie metód, 92% pokrytie príkazov a 97% pokrytie vetvení.

Zaujímavé výstupy experimentu boli doplnené prieskumom, ktorý sa vykonal až na samotnom konci. Oblasti, na ktoré sa pýtali boli efektívnosť, produktivita a náročnosť. Výsledky prieskumu sú zobrazené v Tab. 1.

	% súhlasí	**Otázka**
Produktivita	87.5%	TDD umožňuje lepšie pochopenie požiadaviek
	95.8%	TDD redukuje množstvo ladenia
	50%	TDD znižuje čas potrebný na vývoj
Efektívnosť	92%	TDD prináša vyššiu kvalitu zdrojového kódu
	79%	TDD podporuje jednoduchší dizajn
	71%	TDD technika je zreteľne efektívnejšia
Náročnosť	56%	Princípy TDD sú naročné na osvojenie
	23%	Absencia prvotnej dizajn fázy je prekážkou.

Tab. 1. Výsledky prieskumu o vývoji riadenom testovaním

Záver

Vývoj riadený testovaním prináša vo svete krátkych hraničných termínov a meniacich sa požiadaviek za účelom prispôsobenia sa trhu riešenie, ako tieto požiadavky plniť bez negatívneho dopadu na kvalitu softvéru. Zákazníci si vyžadujú rýchly výstup a zároveň minimálne náklady. Konkurencia firiem zaoberajúcich sa vývojom je taká veľká, že je pre nich existenčne dôležité týmto požiadavkám vyhovieť a tomu musia prispôsobovať svoje procesy. Rozsiahla a čo najkompletnejšia špecifikácia požiadaviek pred samotným vývojom je utópiou, a preto mnohé tímy riadia svoj vývoj postupne sa rozširujúcimi akceptačnými testami, ktoré zároveň zabezpečujú častú spätnú väzbu od zákazníka a robia proces celkovo flexibilnejším. Vývojári majú taktiež možnosť okamžitej kontroly, či je ich súčiastka v danom stave vyhovujúca, čo prispieva kvalite kódu.

Použitá literatúra

1. Bieliková M.: Softvérové inžinierstvo – princípy a manažment, Vydavateľstvo STU, Bratislava (2000), p. 95.
2. Highsmith, J.A.: *Adaptive Software Development*, Dorset House Publishing Company, Inc., New York, 1999.
3. George, B., Williams, L.: An Initial Investigation of Test Driven Development in Industry. In: *Proc. of the 2003 ACM Symposium on Applied Computing*, 2003.
4. Steindl, Ch.: Test-driven development. http://www.agilealliance.org/system/article/file/1423/file.pdf, 2005, p. 7.
5. Jeffries, R.E.: eXtreme Testing. Software Testing & Quality Engineering Magazine, March/April 1999, p. 24.
6. Muller, M. M. and Hagner, O., Experiment about Test-first programming. In: *IEE Proc. – Software*, Volume 149, Issue 5, October 2002, p. 131 – 136.
7. George, B.: *Analysis and Quantification of Test Driven Development Approach*, MS Thesis, North Carolina State University, 2002.

Annotation

One step ahead of errors

Fast and cheap development of high-quality applications is nowadays considered as a standard. Actual software development processes are trying to adapt to these requirements - their phases are defined in a way to minimize potential problems. Wrong understanding of customer's requirements, their instability, errors in software, that all are reasons for development delays and further grow of expenses. Test-driven development together with acceptance tests is a technique trying to prevent these problems and it is getting more and more popular in the field of software development. This essay describes problems with classic approaches of testing and also problems with communication between customer and development teem, introduces test-driven development and shows its advantages by summing up multiple studies. It also presents advantages not often mentioned – psychological and motivational.