PREČO TO NEJDE?

Dobrý test môže odhaliť chybu, ale nikdy nezaručí bezchybnosť!

Samuel Števaňák

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

Abstrakt. *Proces vývoja softvérového riešenia sa skladá z viacerých etáp. Popri analýze, návrhu a implementácii nesmieme zabudnúť na testovanie a s tým spojené zabezpečenie kvality softvérového produktu. Často majú ľudia mylnú predstavu, že proces testovania je jednoduchá alebo nepotrebná činnosť, na ktorú netreba špeciálne vedomosti. Keďže so stúpajúcou zložitosťou rastie množstvo chýb pri vývoji, je dôležité nájsť vhodné metódy a techniky zlepšujúce kvalitu vyvíjaného softvéru, jej meranie alebo zabezpečenie. Preto v eseji opisujem dôležitosť testovania ako samostatnej etapy tvorby softvérového projektu. Dokáže testovanie zaručiť kvalitu, či iba oklame vývojárov o bezchybnosti riešenia? Ďalej rozoberám dopady podcenenia významu testovania počas vývoja, náklady spojené s opravou chýb, opisujem možné problémy, ktoré môžu nastať počas testovania, kedy je vhodné ukončiť testovanie. V poslednej časti porovnám organizáciu testovania vo veľkom tíme a v tíme 6-7 vývojárov.*

Kľúčové slová: *testovanie, dôvody pre testovanie, kvalita softvéru, bezchybnosť,*

Úvod

Kvalita softvérového produktu je veľmi dôležitým atribútom a často pre zákazníka azda najdôležitejším kritériom pri nasadzovaní softvérového riešenia. Myslím si, že v počiatkoch softvérového inžinierstva bol „pánom“ samotný softvér a zákazník iba čakal, čo daný softvér dokáže a v čom mu uľahčí prácu. Postupne, s vývojom softvérového inžinierstva, sa situácia radikálne zmenila. Zmena nastala v tom, že zákazník si úplne

našpecifikuje čo je pre neho dôležité, ako má systém pracovať, na čo sa bude používať a softvérová spoločnosť má za úlohu dané požiadavky kvalitne spracovať a zahrnúť v produkte.

Softvérový priemysel veľmi rýchlo napreduje a s jeho rozvojom sa rozvíja aj konkurenčný boj spoločností o zákazníka. Aby si spoločnosti udržali zákazníkov je potrebné zabezpečiť dostatočnú kvalitu produktu. Vzniká teda tlak na zvýšenie konkurencieschopnosti na trhu a tým aj potreba zvýšenia kvality softvérového riešenia. Dôležité je si uvedomiť, že testovanie, validácia a verifikácia ovplyvňuje výslednú kvalitu výrobku. Ich hlavným cieľom je odhaliť chyby v systéme ešte skôr ako sa výrobok dostane ku zákazníkovi alebo spôsobí určitú škodu.

Testovanie zastáva veľmi dôležitú úlohu v procese vývoja softvéru. Môžeme ho nájsť prakticky vo všetkých fázach vývoja, od plánovania akceptačných testov (špecifikácia systému), plánu testovania jednotiek (hrubý a jemný návrh), regresného testovania počas implementácie až po akceptačné testovanie zákazníkom.

Čo je vlastne tá kvalita?

Podľa ISO 8402 kvalita je súhrn vlastností a charakteristík výrobku, procesu alebo služby, ktoré preukazujú jeho schopnosť splniť určené alebo odvodené potreby. Problém je však v tom, že softvér je nehmotný produkt, čiže neexistujú presné kritéria určenia jeho kvality. Ja sa najviac stotožňujem s názorom, že kvalita softvérového produktu nie je definovaná ako absolútna miera, ale ako stupeň splnenia požiadaviek, resp. potrieb, ktoré si vopred špecifikoval zákazník. Podľa [2] z toho vyplýva aj taký extrém, že ak cieľom bude vyvinúť nefunkčný softvér, potom čím menej bude fungovať, tým bude mať vyššiu kvalitu.

Vyhodnocovanie kvality systému sa môže odlišovať a bežne sa aj odlišuje pri porovnávaní jedného systému s druhým, ale niektoré atribúty ostávajú rovnaké. Týmto atribútom sa konkrétne venuje ISO štandard ISO 9126 a medzi charakteristiky kvality zaraďuje funkcionalitu, spoľahlivosť, použiteľnosť, efektívnosť, udržateľnosť a prenositeľnosť.

Procesy verifikácie a validácie veľmi úzko súvisia s kvalitou softvéru. Sú to akoby určité metódy alebo postupy, ako určiť úroveň kvality daného vyvíjaného produktu. Mali by sme si uvedomiť, že validácia a verifikácia nie sú navzájom synonymá, ale majú úplne iný význam. Verifikácia ma za úlohu určiť, či vyvíjame softvér správnym spôsobom. To znamená, že sa používa na nájdenie a odstránenie chýb počas každej etapy vývoja. Pre každú fázu vývoja dokážeme pomocou verifikácie zistiť, či spĺňa požiadavky, ktoré boli pre ňu určené. Avšak verifikácia nám nezabezpečí, aby výsledný produkt splnil vopred špecifikované požiadavky. Na daný účel slúži validácia. Validácia sa vo väčšine prípadov používa až po skončení etapy vývoja softvéru, aby sme mohli zistiť, či vyvinutý softvér spĺňa požiadavky pre použitie (či sme vytvorili správny výrobok).

Prečo je testovanie dôležité

Ako je uvádzané v [1] testovanie sa dá preložiť ako pozorovanie vykonávania softvérového systému na zistenie, či sa správa ako bolo zamýšľané a na identifikovanie možných chýb. Popri špecifikácii, analýze, návrhu a implementácii má veľmi dôležité

miesto pri vývoji aj etapa testovania. Podľa [4] sa na ňu venuje až 40% všetkých prostriedkov, ktoré boli vyčlenené na daný projekt. Dokonca v prípade, ak je požadovaná vyššia miera spoľahlivosti, bezpečnosti, stability alebo správnosti systému (napríklad systémy vyvíjané pre zdravotníctvo, letectvo, armádu alebo bankovníctvo) môže byť na etapu testovania vyčlenených viac ako 50% prostriedkov.

V dnešnej dobe sme svedkami vývoja mnohých zložitých softvérových nástrojov, ktoré vyžadujú spoluprácu viacerých vývojárov. Je potrebné neustále reagovať na požiadavky užívateľov a vyvíjaný softvér rozširovať a optimalizovať. Dochádza tak k častým zmenám zdrojového kódu a ďalších súčastí softvéru. Systémy sa stávajú rozsiahlejšími, kód menej prehľadným a horšie zrozumiteľným. Jednotliví vývojári zdieľajú veľkú časť zdrojového kódu, zasahujú do rôznych častí projektu, modifikujú kód, ktorý sami nevytvárali. Ešte k tomu sú často ovplyvnení krátkymi termínmi dokončenia. V takom prostredí nutne dochádza k vytváraniu chýb vo vyvíjanom softvéri, preto vzniká potreba chyby nájsť a odstrániť.

Ideálnym cieľom vývoja softvéru je dosiahnuť nulovú chybovosť produktu. Prax je však iná. Hovorí sa, že v každom softvéri je minimálne jedna chyba. Napriek tomu, že programátori často považujú testovanie za zdĺhavú a o čas oberajúcu činnosť. Myslím, že dobre navrhnuté testy môžu odhaliť veľké množstvo chýb, ktoré sa následne dajú redukovať na minimálnu úroveň. V nasledujúcich bodoch uvediem niekoľko ďalších dôvodov, prečo je dôležité softvér testovať:

- Overenie, že všetky požiadavky boli analyzované správne. Veľa zo závažných zlyhaní softvéru bolo zapríčinených nekompletnými alebo chýbajúcimi požiadavkami. Testovanie overuje, či sú požiadavky relevantné, koherentné, merateľné, kompletné a testovateľné.
- Overenie, že všetky požiadavky boli implementované správne. Adekvátne testovanie zabezpečuje, že softvér pracuje tak, ako sa očakáva. Na jednotlivé dopyty používateľa odpovedá správne a pracuje podľa zadanej špecifikácie.
- Identifikovanie chýb v skorších fázach vývoja produktu. Možno väčšina z nás vie, že čím skôr sa chyba v systéme odhalí, tým menej stojí jej odstránenie.
- Verifikovanie interakcií medzi softvérovými súčiastkami, lebo množstvo chýb v systémoch vzniká práve pri integrovaní častí systému. Vhodným návrhom testov môžeme tieto chyby identifikovať a následne odstrániť
- Zvyšovanie spoľahlivosti aplikácie.

Mám taký dojem, že testovanie je niekedy brané ako iba posledná činnosť pri vývoji softvéru, ktorá sa vykonáva po ukončení programovania produktu. Namiesto toho, testovanie by malo byť vykonávané v každej etape vývoja. Ak rozdelíme životný cyklus vývoja softvérového riešenia do analýzy požiadaviek, návrhu, implementácie, nasadenia a údržby, potom testovanie by malo sprevádzať každú z týchto fáz. Ak je izolované iba ako samostatná fáza, ktorá sa vykonáva neskoro v životnom cykle softvéru, chyby v analýze alebo návrhu môžu vyústiť do nenormálnych rozmerov a náklady na ich opravu môžu prevýšiť vyčlenený rozpočet na projekt. Kvôli tomu by testovanie nemalo byť izolované ako kontrolná činnosť na záver projektu.

Zaručí testovanie bezchybnosť?

V novinách sa môžeme pravidelne dočítať o systéme, ktorý pod náporom používateľov skolaboval (napr. Internerbanking hneď niekoľko českých bánk, elektronický obchod nemenovanej slovenskej spoločnosti zameriavajúcej sa na e-business). Dopad týchto a podobných výpadkoch môže zapríčiniť nevyčísliteľné finančné straty, poškodenie dobrého mena spoločnosti a v neposlednej rade úsilie spojené s investíciami vedúcimi k náprave do požadovaného stavu. Vo vyššie spomínaných problémoch išlo podľa mňa o podcenenie záťažového testovania (resp. výkonnostného testovania). Pri jeho dôkladnejšom použití by bolo možné odhaliť správanie sa systému pri veľkom počte transakcií v systéme.

Existujú ľudia, ktorí neveria v existenciu bezchybného softvéru a taktiež existujú ľudia, ktorí si myslia, že testovanie bezchybnosť zaručí. Ja som toho názoru, že neexistuje bezchybný softvér. Samozrejme mám na mysli iba zložitejšie softvérové systémy. Bezchybné programy existujú ale iba tie triviálne. Napríklad nám všetkým dobre známy program, ktorý umožní vypísať na obrazovku „Hello word!“ je možné vytvoriť bez chyby a verím, že sa to mnohým aj podarilo. Čiže podľa predošlej úvahy som dospel k tomu, že testovanie nám nedáva istotu, že softvér je naozaj bezchybný. Som zástancom názoru, že fungujúci program má iba neobjavené chyby. To znamená, že aj keď navrhnuté testy neobjavili žiadne nové chyby v etape overovania produktu, program napriek tomu obsahuje chyby. Chybu je možné hľadať v nepresne napísaných testovacích scenároch alebo nepokrytí všetkých scenárov použitia, požiadaviek alebo samotnej funkcionality.

Existuje test na test?

Dostal som sa ku otázke, či sa dá overiť, že navrhnutý test je správny, resp. či existuje test na test. Pri otestovaní softvéru bez nájdenia jediného problému môžu podľa mňa nastať dva prípady:

- Testovaný softvér je naozaj bezchybný
- Testy sú chybné

Prvá možnosť je vo väčšine prípadov najmenej pravdepodobná kvôli tomu, že takmer žiadny softvérový produkt nie je bez chýb. Takmer nikdy nie je možné otestovať všetky časti softvéru na 100 %. Ako teda zistiť, že navrhnuté a používané testy sú správne? Nepodarilo sa mi nájsť odpoveď na túto otázku v žiadnej vedeckej literatúre. Ale napriek tomu existujú návody, ktoré radia ako vytvárať kvalitné testy a ako sa vyhýbať chybám. Najviac sa prikláňam k myšlienke, že najlepšie je použiť testovacie prípady (eng. test case) na opis, ako daný systém funguje. Pri použití spomenutej myšlienky zistíme, že testovacie prípady by sa mohli používať ako „dokumentácia“ opisujúca, ako daný systém pracuje a používateľ by bol oboznámený, ako má používať používateľské rozhrania a aké druhy chýb môže očakávať. Myslím, že taktiež je dôležité, aby testy pokryli čo možno najviac funkcionality systému alebo požiadaviek používateľa.

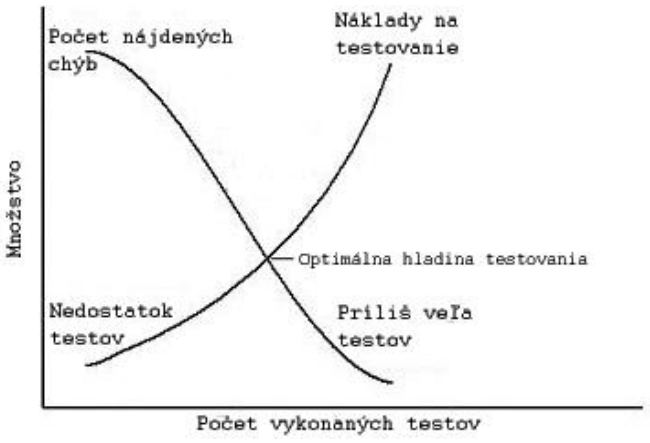
Kedy prestať testovať

Myslím, že je veľmi ťažké sa rozhodnúť, kedy je už naozaj všetko potrebné otestované a môžeme túto etapu ukončiť. Množstvo moderných softvérových aplikácií je príliš zložitých, bežia v rôznych navzájom závislých prostrediach, takže v takých prípadoch je nemožné uskutočniť kompletné otestovanie. „Kedy naozaj ukončiť testovanie?“ je jedna z najťažších otázok pre testovacieho inžiniera. Myslím, že najdôležitejšie faktory na zastavenie testovania sú nasledujúce:

- Dodržiavanie termínov, kedy je potrebné odovzdať celý produkt v stanovený čas, prípadne treba dodržať termín dokončenia testovania aby mohli prebiehať ďalšie časti vývoja.
- Testovacie prípady sú splnené v určitom relatívnom množstve, ktoré postačuje pre fázu testovania.
- Prostriedky, ktoré boli vyčlenené pre testovanie sa minuli.
- Nájdenie nejakej chyby je príliš pomalé. To znamená, že ak sa chyby už nevyskytujú alebo iba veľmi zriedka, tak je možné ukončiť etapu testovania, aj keď s určitým rizikom.
- V prípade, že je riziko v projekte pod akceptovateľným limitom, je taktiež možné ukončiť testovanie.

Domnievam sa, že rozhodnutie na zastavenie testovania je založené na úrovni akceptovaného rizika, ktoré určil manažment. Keďže testovanie je nekonečný proces, nikdy nemôžeme zaručiť, že bolo vykonané testovanie na 100 %. Dokážeme iba minimalizovať riziko odovzdania produktu zákazníkovi, na ktorom bolo urobených nedostatočný počet testov.

Patton [3] vo vzťahu ku obrázku 1 tvrdí, že každý softvérový projekt má určitú efektívnu hladinu testovania. Myslím, že je to pravda, ale v značnej miere zjednodušená. Situácia sa komplikuje neuniverzálnosťou testovania. Ako som už spomínal, každý produkt je jedinečný a testy sú navrhované na základe skúseností a preferencií toho, kto o ich použití rozhoduje. Takže dvaja ľudia môžu pre jeden produkt navrhnúť dve rôzne sady testov s rôznymi nákladmi.

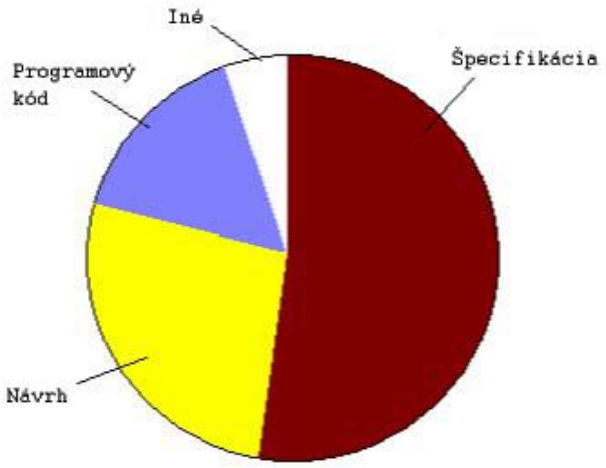


Obr. 1. Optimálna hladina testovania [3]

Kedy vzniká najviac chýb?

Podľa Pattona [3] je chyba čokoľvek ohľadne programu, čo podľa niektorého používateľa alebo vývojára znižuje hodnotu programu. Ja osobne sa taktiež prikláňam k tejto definícii, pretože platí aj v prípade, keď program nie je úplne kompletný, čo patrí taktiež medzi chyby. Skutočne dôležité je vedieť, kde sa chyby najčastejšie vyskytujú a kde je možné objaviť tie najdôležitejšie, ktorých oprava by v neskorých fázach projektu mohla byť veľmi drahá. Práve na tieto miesta je potrebné sa najviac zamerať.

Väčšina materiálov o testovaní, ktoré sa mi dostali do rúk, uvádza obrázok 2. Daný obrázok znázorňuje, že viac ako polovicu chýb má na svedomí špecifikácia, hoci moje skúsenosti tomu dostatočne nedávajú za pravdu. Je pravdepodobné, že niekedy toto rozdelenie vzniku chýb bolo úplne presné, ale s pokrokom softvérového inžinierstva sa toto rozdelenie postupne menilo. Domnievam sa, že aj keď tie najťažšie objaviteľné a s najväčším dopadom sú práve chyby majúce pôvod v špecifikácii či návrhu, najčastejšie sú chyby programového kódu alebo vzhľadu aplikácie. Tie činia 40-60 %. Ale taktiež je zrejmé, že daná charakteristika závisí na type projektu. Vývoj riadený testami (Test driven development) alebo schopný analytik či skúsení programátori môžu percenta chýb podľa pôvodu značne ovplyvniť.



Obr. 2. Rozdelenie chýb podľa ich vzniku [3]

Myslím, že prvá vec, ktorá si zaslúži značnú pozornosť, je práve špecifikácia a návrh. Prečo? V prvom rade kvôli dopadu týchto chýb. Pre väčšinu z nás je príliš pomalý softvér oveľa horší ako nejaký preklep na jednom z formulárov. Ďalším dôvodom venovania pozornosti špecifikácii a návrhu je, že na základe týchto dokumentov si testeri môžu zistiť požiadavky zákazníka a zoznámiť sa s vyvíjaným softvérom. Podľa spomenutých dôvodov môže sústredenie sa na tieto časti vývoja ušetriť nepríjemné prekvapenia a najviac ovplyvniť výnosnosť projektu.

Ako je to s testovaním v malých a veľkých tímoch

Na väčších projektoch sa testeri združujú do jedného prípadne viacerých testovacích tímov, v rámci ktorých majú špecifické úlohy. Samozrejme podmienkou je dostatok prostriedkov, najmä ľudských, preto sa testovacie tímy používajú hlavne vo veľkých spoločnostiach. Veľkou výhodou testovacích tímov je ich nezávislosť a špecializácia. Nepodieľajú sa priamo na vývoji, a preto pri testovaní môžu použiť úplne iný pohľad na problém, ako by mohol mať napr. programátor. Myslím, že vo všeobecnosti by ten istý človek nemal testovať vec, ktorú sám urobil. Hlavou projektu je project manager, ktorý koordinuje projekt a tímy. Team manager má na starosti chod prideleného tímu. Team lead riadi a rozdeľuje úlohy v rámci tímu, pripravuje prípady testovania a testovacích skriptov. Tester má za úlohu testovať a zaznamenávať výsledky testov.

Opisoval som prevažne testovanie vo veľkých projektoch, ktoré mali k dispozícii samostatný testovací tím, poprípade viac tímov. Je možné použiť spomínané metódy na malých projektoch o veľkosti šesť až sedem ľudí (tímy, ktoré riešia školské zadanie)?

Hneď ako prvé je potrebné uviesť, že malý projekt má veľké obmedzenie na strane zdrojov, v tomto prípade ide hlavne o ľudské zdroje. Naopak, výhodu ma malý tím v tom, že nie je potrebné sa zaoberať zložitou réžiou a manažmentom veľkej skupiny osôb. Najväčším problémom ostáva rozdelenie úloh v rámci tímu, ako je manažment kvality, manažment rizík, manažment plánovania, vývoj... Napadá ma iba jediné možné riešenie, ktoré spočíva v zlučovaní, inak povedané prekrývaní úloh. Podľa tejto úvahy sa predpokladá, že každý člen tímu sa bude podieľať na vývoji produktu a taktiež sa okrem toho bude snažiť robiť jednu z vyššie spomínaných funkcií.

Ako som už spomínal, celý proces testovania by mal mať na starosti jeden člen tímu, ktorý bude samozrejme podporovanými ostatnými členmi hlavne vo fáze vykonávania testov. Za úlohu bude mať plánovať testovanie, pripravovať a vykonávať testy a vytvárať testovaciu dokumentáciu.

Ak sa pokúsim o porovnanie úloh vo veľkom a malom tíme dostanem nasledujúce možnosti:

- Testovací tím v prípade malého projektu predstavuje jediná osoba, ktorá je zodpovedná za kvalitu a testovanie. V prípade potreby môžu testovací tím tvoriť aj ďalší členovia tímu.
- Všetky časti testovacej dokumentácie sú aplikovateľné v rovnakej miere v malom aj veľkom tíme. To znamená, že dokumentácia nezávisí od veľkosti tímu.
- Testovacie prostredia je možné používať aj v prípade malých tímov, hoci ich výsledný prínos nie je až taký veľký ako v prípade veľkých projektoch. Myslím, že zlepšenie, dosiahnuté použitím testovacieho prostredia, nie je až také zreteľné.

Metodika a modely vývoja softvérových riešení, ktoré zahŕňajú aj manažment testovania, sú definované univerzálne a sú ľahko prispôsobiteľné potrebám konkrétneho projektu.

Záver

Testovanie sa často, hlavne v menších spoločnostiach a na rozsahovo menších projektoch , dostáva do úzadia. Je to spôsobené aj tým, že si niekedy celkom neuvedomujeme jeho

zložitosť. Etapa testovania je nevyhnutná v každom vývoji softvérového riešenia, či už ide o malý alebo veľký projekt. Hrá významnú úlohu pri posudzovaní kvality. Znázornil som, že veľkou chybou je podcenenie testovania v prvých etapách životného cyklu softvéru. Neskoro objavené chyby môžu mať za následok veľké výdavky na ich odstránenie a tým aj zdržanie celého projektu. Výsledkom správne zvoleného testovania je zvýšená kvalita produktu a tým aj vyššia úspešnosť produktov na trhu a dobré meno spoločnosti.

Použitá literatúra

1. Bertolino, A.: Software Testing Research: Achievements, Challenges, Dreams, IEEE Computer Society, 2007.
2. Bieliková, M.: Manažment v softvérovom inžinierstve, Vydavateľstvo STU, Bratislava, 1999.
3. Patton, R.: Testovaní softwaru, Computer Press, Brno, 2002.
4. Saha, G.M.: Understandig software testing concepts, ACM Ubiquity Vol.9, Issue6, 2008.

Annotation

Why does not it work?

A software solution development process consists of more periods. With analysis, design and implementation of a software product, testing together with quality cannot be missed out. It is a common point of view that testing is a simple or unnecessary activity, that doesn't demand any special knowledge. However, with growing complexity grows number of errors in development process. Therefore, it is vital to find suitable methods and techniques that would improve measuring or ensuring of quality of the developed software. This is the reason why I describe importance of testing as an independent part of a software project development process. Does testing ensure quality or just misleads developers about correctness of their product? Next, I look on effects of underestimating the importance of testing during the development, cost of correction and I describe possible problems that can occur during the testing process and when to stop testing. Finally, I compare organization of testing in large team to team of 6-7 developers.