VÝHODNOSŤ PREVENCIE

Preventívne zistime, či sa prevencia oplatí!

Ondrej Kuzmík

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

Abstrakt. *Testovať či netestovať? Ak áno, tak kedy, čo a ako? Možnosť vytvorenia eseje využijem pre opísanie svojho názor na túto problematiku alebo aspoň vybranú časť. Priblížim svoj pohľad na tému testovania a kvality. Kedy sa testovanie oplatí, a kedy je viac menej zbytočné. Väčšinou sa testuje správnosť kódu, pričom na testovanie požiadaviek (splnenia funkcionality) sa zabúda. Túto problematiku sa pokúsim priblížiť, spolu s tým, ako môže kvalitné testovania dopomôcť ku lepšiemu softvéru, prípadne spraviť softvér výnimočným. Zameriam sa aj na tému, kto by mal byť za testy zodpovedný a ako je tento človek dôležitý pre tím.*

Kľúčové slová: *kvalita softvéru, testovanie, tímový hrdina*

Úvod

Je nepopierateľné, že testovanie dokáže zlepšiť výslednú kvalitu produktu. Nič však nie je zadarmo. Pre kvalitné testovanie je potrebné obetovať zdroje, ako napríklad čas, prípadne financie. Preto treba zvážiť, či je testovanie pre vývoj produktu naozaj výhodné. V eseji sa zameriam na problematiku výhodnosti či nevýhodnosti testovania, oproti jednoduchšej oprave prípadných chýb.

Existujú rôzne oblasti testovania a nie vždy majú rovnaké kritéria. To znamená, že produkt môže byť výhodné testovať len istú časť funkcionality a to jedným, konkrétnym, spôsobom. Po hlbšom zamyslení sa nad týmto problém vystáva otázka, kto má byť za rozhodnutia okolo testovania zodpovedný. Stačí jeden človek alebo má byť zodpovedný celý tím? Priblížim svoj názor aj na tento problém.

Prevencia alebo liečba – programátorské chyby

Čo majú spoločné? Spoločné majú to, že riešia problémy. Rozdielne majú, kedy tieto problémy riešia. Pod prevenciou si predstavme, že by sme sa dali zaočkovať. Liečba znamená, že ležíme v nemocnici a tam nás dávajú do poriadku. Asi väčšine ľudí, vrátane mňa, sa prevencia zdá sympatickejšia. Čo ak ale prevencia zlyhá? Zaočkujú nás pokazenou vakcínou a následne na danú chorobu ochorieme (z vakcíny), prípadne očkovanie nezaberie a choroba sa dostaví tak či tak. Stále vyzerá prevencia sympatickejšia? Myslím, že áno, ale už nie tak výrazne.

Spomínal som to preto, že pri vytváraní softvéru sú dve hlavné možnosti, ako zlepšiť jeho kvalitu testovaním. Už asi tušíte, že sa jedná o prevenciu a liečbu. Ako píše R. Geoff Dromey [2], tak liečba spočíva vo vytvorení softvéru, následnom aplikovaní testov ako aj otestovaní používateľmi. Po tejto fáze sa nájdené chyby opravia a vznikne novšia verzia softvéru. Prevencia, ktorú R. Geoff Dromey preferuje, spočíva v prvom rade vo vytvorení takzvaných „quality-carrying properties“, ktoré môžu predstavovať napríklad jednotlivé testy, hlavne pri vývoji riadenom testami. Taktiež je potrebné vytvorenie prototypu a používanie rôznych nástrojov, aby sa zaistilo, že „quality-carrying properties“ zabránia vytváraniu chýb.

Ako prvé, čo mi napadne je časová náročnosť kvalitnej prevencie. Je všeobecne známe a logické, že pokiaľ sa chyba objaví skôr, tak nás stojí jej oprava menej času a zdrojov, čo je pre výslednú kvalitu prospešné.

Zamyslime sa však, čo nás toto odhalenie stojí. Pri menších projektoch sa môže stať, že nám vytvorenie kvalitnej prevencie zaberie väčšie množstvo času, ako vytvorenie samotného produktu. Mal som možnosť vidieť, aspoň v menšej miere, ako vývoj riadený testami funguje. Predstavme si menšiu webovú stránku. Nie Facebook alebo Twitter, ale napríklad stránku Slovenskej národnej galérie, kde sa rozhodneme dorobiť prihlásenie. To znamená pridanie tlačítka, pár riadkov kódu pre overenie, prípadne cookies. Treba nám pri tejto úlohe kvalitnú prevenciu? Nestačí nám otestovať si prihlásenie ručne a mať po probléme? Kvalitné testy by nám zabrali rovnako veľa, ak nie viac času, ako napísanie výsledného kódu. Dokonca je možné alebo skôr veľmi pravdepodobné, že sa pri písaní kódu nepomýlime a testy by tým pádom nič neodhalili. Myslím si, že ak v tomto prípade zahrnieme do kvality aj časovú náročnosť, tak jednoduchá liečba (ak by sa vyskytla chyba) by bola lepšia.

Zamysleli sme sa nad menej náročnými projektmi, kde sa často môže stať, že nám prevencia zaberie viacej času, ako samotný projekt. V týchto projektoch je možnosť výskytu chyby a jej následný dopad na čas a zdroje menší. Na rad prichádza náročnejší projekt. Napríklad komerčný e-shop. Ak sa jedná o projekt pre zákazníka, tak predstava liečby začína byť menej sympatická. Zákazník by chcel ideálne dostať hotový, funkčný a bezchybný produkt. Ide taktiež o väčší projekt, kde je možnosť chyby a následná cena opravy vyššia. Stále je ale možnosť vytvorenia projektu, ktorý by mal minimálnu chybovosť a následná liečba by nebola problémom, v niektorých prípadoch sa už ale využitie prevencie stáva výhodnejším.

Projekt typu Facebook alebo Twitter, si už ale bez kvalitnej prevencie predstaviť neviem. Pri projekte takejto náročnosti, na ktorom môže pracovať viacej vývojových tímov,

je zavedenie kvalitných testov, ktoré môžu odhaliť problémy fungujúcich častí, ktoré vznikli pridaním novej funkcionality, mi príde ako nevyhnutnosť. Ak by sa problémy odhalili, až po zavedení funkcionality do prevádzky, tak by ich oprava a náprava problémov, ktoré spôsobili bola veľmi náročná.

Podľa môjho názoru je teda potrebné pozrieť sa na veľkosť projektu, prípadne programátorské schopnosti a následne si vybrať či použijeme prevenciu, alebo sa spoľahneme na liečbu.

Prevencia alebo liečba – použiteľnosť

Zamyslime sa najprv, ako si v tomto prípade môžeme prevenciu, respektíve liečbu predstaviť. Kam presne spadá alfa, respektíve beta testovanie? Podľa mňa to závisí od podmienok. Ak vypustíme produkt do otvoreného beta testovanie, s tým, že ide už o takmer hotový produkt, kde je informácia o tom, že prebieha beta testovanie, len niekde v pozadí, tak prípadne opravy považujem za liečbu. Za liečbu to budú podľa mňa považovať aj používatelia a v prípade zlého dojmu si môžu o produkte vytvoriť negatívny názor. Preto je v tomto prípade potrebné postupovať opatrne a dať si na beta verzii výrazne záležať.

Kedy sa ale vôbec toto testovanie oplatí? Podľa mňa tu už nie je rozhodujúcim prvkom komplikovanosť funkcionality z pohľadu programátorských chýb, ako to bolo v predošlej kapitole. Skôr ide o komplikovanosť funkcionality z používateľského pohľadu. Či je funkcionalita dostatočne intuitívna, prípadne zaujímavá pre používateľa.

Najprv by som sa chcel zamerať na svoj názor pre problém intuitívnosti. Zoberme si ako príklad problém vytvárania e-shopu. Na čo sa v tomto prípade testovania použiteľnosti treba zamerať? A je vôbec testovanie potrebné? Podľa mňa určite. Pokiaľ by bol priebeh nákupu v spomínanom e-shope navrhnutý zle, tak by sa mohlo stať, že by používateľ o nákup stratil záujem, prípadne by ho nedokázal zrealizovať. Čo ale presne treba testovať? Tu sa dostávame ku problému komplikovanosti funkcionality z pohľadu používateľa. Ako som spomínal tak ja by som testovanie využil napríklad pre proces nákupu tovaru. Od začiatku až po odoslanie potvrdzujúceho formulára. Tento proces by som testoval nielen pred prvotným uvedením e-shopu do prevádzky, kedy býva testovanie oveľa rozsiahlejšie, ale taktiež pri prípadných úpravách procesu v už fungujúcom e-shope.

Taktiež ale nie je potrebné využívať prevenciu všade. Príkladom môže byť proces prihlásenia. Aj keď pri úvodnom testovaní produktu, pred nasadením do prevádzky, môžeme okrajovo otestovať aj tento proces, tak pri jeho úprave, by som z pohľadu celkovej náročnosti, nechal odhalenie prípadných používateľských problémov na liečbu.

Ako ale problém rozdeliť z pohľadu záujmu používateľa? Tu sa treba zamyslieť nad tým, či bude výsledný produkt určený dostatočne veľkej skupine používateľov, aby bolo výhodne ho dopredu testovať. Ak by produkt malo používať iba pár ľudí, liečba podľa mňa vyznieva oveľa prijateľnejšie, ako ak by produkt malo používať veľké množstvo ľudí.

Prevencia alebo liečba – požiadavky

Pokiaľ sa priemerného programátora opýtame na čo slúži testovanie, tak nám pravdepodobne odpovie, že na odhalenie chýb v kóde. Na tejto odpovedi nie je nič prekvapivé. Je ale toto naozaj to, na čo sa je potrebné pri testovaní zamerať? Nasib S. Gilla v svojom článku [3] navrhuje zamerať sa taktiež na testovanie požiadaviek.

V predchádzajúcich kapitolách som sa zameral na testovanie funkcionality. Z toho vyplynulo, že potreba prevencie bola závislá na veľkosti projektu, respektíve zložitosti funkcionality. V prípade testovania požiadaviek to však vidím inak.

Predstavme si, že vytvárame produkt, ktorý si od nás objednal zákazník. Samozrejme, že je potrebné aby bol výsledný produkt v očakávanej kvalite, ale rozhodujúcim faktorom spokojnosti zákazníka bude predovšetkým splnenie jeho požiadaviek.

Okrem požiadaviek, ktoré nám zákazník zadá, vedia podľa mňa výsledný dojem ovplyvniť najmä požiadavky, ktoré zákazník nezadá, prípadne o nich ani nevie, ale produkt ich bude obsahovať. Práve toto dokáže produkt a vývojový tím posunúť v očiach zákazníka na vyššiu úroveň.

Preto by som podľa mňa rozdelil požiadavky na predpokladané a „niečo navyše“. V dnešnej doba sa často na konci vývoja softvéru spýtame: „Je tam všetko, čo od nás očakávajú?“. Toto v mnohých prípadoch postačuje. Ak ale chceme vytvoriť niečo výnimočné, prípadne niečo čo by mohlo vyniknúť, treba sa opýtať „Naozaj tam už nemôžeme nič lepšie vytvoriť?“.

Z pohľadu prevencie, respektíve liečby by som v tomto prípade rozhodne uprednostnil prevenciu. Pýtať sa neustále čo sa dá na produkte vylepšiť ešte pred jeho konečným odovzdaním, má podľa mňa oveľa väčší dopad, ako jeho prípadné vylepšovanie po odovzdaní. To z dôvodu, že postupné zlepšovanie produktu po jeho odovzdaní je síce možné, ale nemá už taký „očarujúci“ efekt, aký by vznikol pri prvom pohľade na produkt. Taktiež aj motivácia a nadšenie tímu môžu časom klesať.

Kvalitár – hrdina

Kto ja ale nakoniec za rozhodnutie o tom či použiť prevenciu, alebo liečbu zodpovedný? Ako som spomenul tak vidím niekoľko rôznych spôsobov testovania, ktoré sú výhodné do rôznych produktov, respektíve ich častí. J. Bach vo svojej úvahe tvrdí [1], že dôležitejšie ako rôzne spôsoby vývoja, sú pre projekt dôležití ľudia, ktorí na ňom pracujú. Do popredia dáva skutočnosť, že tím potrebuje hrdinu, ktorý na seba dokáže zobrať zodpovednosť a riziká.

Podľa mňa by malo ísť o kvalitára. Niekto môže tvrdiť, že za tím, a výsledný produkt, má najväčšiu zodpovednosť vedúci tímu. Podľa mňa je však najväčšia zodpovednosť na kvalitárovi. Z môjho pohľadu sa dá zodpovednosť rozdeliť na vonkajšiu (vzhľadom k ľudom, zákazníkom) a vnútornú v rámci tímu. Navonok má zodpovednosť vedúci tímu, ale vnútorne kvalitár, ktorý sa musí rozhodnúť ako zabezpečiť, aby bol výsledný produkt kvalitný. Keď si teraz porovnám tieto dve zodpovednosti, tak si myslím, že vnútorná zodpovednosť je dôležitejšia.

Záver

Objasnil som môj pohľad na rôzne spôsoby testovania a opravy chýb. Ak by som si mal vybrať, či použiť prevenciu, alebo liečbu pre všeobecný projekt, tak by som odporučil prevenciu. Treba si však zhrnúť všetky vlastnosti projektu, ako sú jeho veľkosť, cieľová skupina a iné a následne po dôkladnom zvážení vybrať lepší prístup. Aj potom sa ešte treba rozhodnúť, či je vybraný prístup lepší v každom prípade, alebo sa použije pre každú časť projektu iný. Práve v tomto by mohla moja esej čitateľovi, ideálne kvalitárovi, pomôcť. Preto moja záverečná rada znie: „Preventívne zistime, či sa prevencia oplatí!"

Použitá literatúra

1. Bach, J.: Enough About Process: What We Need Are Heroes. *Computer*, Volume 12, Number 2 (1995), 96-98.
2. Dromey, R.G.: Software Quality—Prevention versus Cure? *Software Quality Journal*, Volume 11, Number 3 (2003), 197-210.
3. Gill, N.S.: Factors affecting effective software quality management revisited. *ACM SIGSOFT Software Engineering Notes*, Volume 30, Number 2 (2005), 1 – 4.

Annotation

The Convenience of Prevention

Test or not to test? If so, when, what and how? I use this opportunity to write an eassy to describe my opinion on the matter or at least a part of it. I describe my view on the connection between testing and quality. When testing is worthwhile and when it is not. Tests are usually made to reveal bugs in code and testing the requirements and functionality is often neglected. I try to bring my point on this issue, along with how good testing can make software better or oustanding. I also focus on who should be responsible for testing and the importantance of this person for the team.