MALÝ PROJEKT NETESTUJ!

Testovať je potrebné len tie veci, ktoré chceme aby fungovali.

Bc. Michal Dávid

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
xdavid[zavináč]is[.]stuba[.]sk

Abstrakt. *Testovanie je súčasťou každého projektu, či už menšieho alebo väčšieho, softvérového alebo aj nesoftvérového. Mnohými býva podceňované a v mnohých prípadoch je zastrčené niekde na podradnom mieste v projekte, väčšinou na tom poslednom. Keď je také dôležité, položme si teda otázku, prečo má v mnohých prípadoch takú nelichotivú nálepku s nápisom "Nepodstatné!"? Podľa mňa hlavne malé projekty testovanie vo veľkej miere stavajú na vedľajšiu koľaj, čo vedie v najlepších prípadoch len k strate času stráveného na projekte. Malý projekt netestuj! Kto povie takúto vetu nie je hodný práce na akomkoľvek projekte. Každý malý projekt môže v sebe skrývať záludné miesta, ktoré v konečnom dôsledku môžu mať veľký význam. V eseji poukážem na to, že testovanie akéhokoľvek malého projektu má zmysel a pokúsim sa ho zbaviť spomínanej nálepky „Nepodstatné!". Taktiež objasním, že testovanie nie je v malých projektoch na predlžovanie procesu tvorby ale v mnohých prípadoch na záchranu celého projektu.*

Kľúčové slová: *testovanie, kvalita*

Úvod

Pri vytváraní projektu sa človek stretáva s mnohými otázkami, na ktoré často krát je veľmi ťažké nájsť odpovede. Nejedná sa ani tak o samotný obsah toho, čo vytvárame. Tam je hľadanie odpovedí priamo určené, pretože spravidla vytvárame niečo, čo ešte neexistuje. Ale jedná sa o samotný proces vytvárania. Samotný životný cyklus. Otázku ktorú sa budem snažiť zodpovedať ja, je: „Načo testovať malý projekt?"

Mnohí sú totiž presvedčení, že testovanie malých projektov je nepotrebné a všetko sa dá pri takom „jednoduchom“ probléme odhaliť hneď pri programovaní. Iní zas považujú testovanie len za akési umelé predlžovanie životného cyklu projektu. A poslednou skupinou sú tímy, kde na testovanie jednoducho nevydá čas.

Všetky tieto skupiny majú ale jedno spoločné. Obrovské riziko, že to čo vytvorili nie je to čo zákazník požadoval. S tým súvisí strata dôvery zákazníka, strata finančných prostriedkov, keďže zákazník predsa nezaplatí za niečo, čo nechcel, a v neposlednom rade aj poškodenie dobrého mena spoločnosti a na to nadväzujúce nepríjemné udalosti.

Malé projekty sú veľmi zradné už vo svojej podstate. Evokujú totiž pocit, že problémy, ktoré sa v nich riešia sú jednoduché a ich riešenie je v podstate priamočiare a nevyžaduje veľkú pozornosť. Veľké projekty naopak vzbudzujú rešpekt a preto sa k nim staviame zodpovednejšie, sme si vedomí dôležitosti a náročnosti problémov už v zárodku. A ako mnohí začínajúci vývojári, ani ja som nebol výnimkou. Plný elánu som sa pustil do jednoduchého projektu a ... Ako to dopadlo?

Čo je testovanie?

Testovanie, aj keď si to možno neuvedomujeme je všade okolo nás. Keď si ráno nalejem pohár čistej vody, tak si ju prezriem, či je čistá a či je teda vhodná na pitie. Je to len veľmi jednoduchý test, ale bez zrakového skontrolovania by som sa pokojne mohol napiť špinavej vody alebo dokonca by to mohla aj iná kvapalina ako voda.

Pri testovaní sa deje presne to isté hoci väčšinou na hlbšej úrovni. Aj pri na naliatí spomínaného pohára sa najprv spýtam: „Je to to, čo som naozaj chcel?“. Podvedome som očakával čistú vodu, ktorej sa môžem bez problémov napiť. Prvou formou testu je vizuálne skontrolovanie. Pokiaľ „test“ neprebehne úspešne nepokračujem ďalej a skúsim danú skutočnosť opraviť. Presne taká istá otázka sa kladie pri testovaní v projekte. „Je to, čo som vytvoril, naozaj to čo zákazník požadoval?“

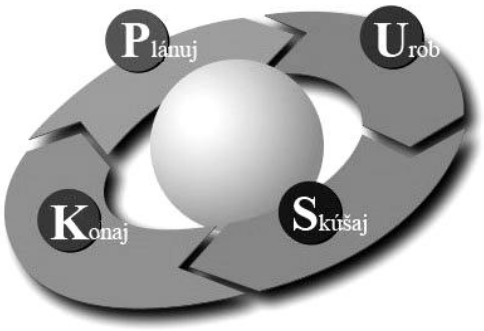
Testovanie systému je overovanie zhody so špecifikovanými požiadavkami. Podľa [1] sú základné druhy testovania nasledujúce:

- Akceptačné testovanie – sleduje sa správanie systému a porovnáva sa s požiadavkami zákazníka.
- Alfa testovanie – testujú pracovníci z organizácie, ktorá tento softvér vyvinula, a podávajú správy o funkčnosti.
- Beta testovanie – veľmi podobné alfa testovaniu s rozdielom, že testuje široká skupina ľudí spravidla nie z organizácie, ktorá systém vyvinula.
- Regresné testovanie – opätovné testovanie systému po zmene nejakého komponentu, prípadne po odstránení nejakej komplexnejšej chyby.
- Výkonnostné testovanie – testovanie na overenie výkonnostných požiadaviek
- Záťažové testovanie – testovanie systému pri jeho maximálnej záťaži a následné pozorovanie jeho správania.

Keďže testovanie je vykonávané za účelom nájdenia chýb, je potrebné pre úplnosť uviesť techniky na ich odhaľovanie. Techniky sú založené na mnohých faktoroch ako sú skúsenosti programátorov a vývojárov, štruktúra kódu, špecifikácia, oblasti použitia softvéru a mnohé ďalšie. Podľa [4] sú to dve základné metódy.

- Metóda čiernej skrinky – neberie sa do úvahy implementačná časť ale len vstupy a výstupy zo systému, ktoré sa porovnávajú s očakávanými.
- Metóda bielej skrinky – potrebná znalosť návrhu a implementácie, vstupy sa vyberajú na základe preskúmavania štruktúry systému.

Pri každej činnosti v projekte sa prechádza niekoľkými etapami. Podľa [2] (Obr.1.) je prvou etapou plánovanie, kde sa určí čo a kedy sa bude na danej časti projektu vykonávať. Druhou etapou je samotné vykonanie danej časti. Následne sa pokračuje pre mňa hlavnou časťou, testovaním. A následne sa otestovaná časť analyzuje. V prípade, že sa vyskytnú odchýlky od plánu je potrebné vykonať kroky na zlepšenie a zároveň, ak je to možné, sa podobným chybám vyvarovať v budúcnosti.



Obr. 1. Shewhart - Demingov kruh [2]

Testovanie a kvalita softvérového projektu

Testovanie v takom rozsahu ako som priblížil naliatím si pohára vody je samozrejme veľmi jednoduché a takmer sa ani nedá prirovnávať k testovaniu softvérového produktu. Projektom sa rozumie časovo ohraničené obdobie s cieľom vytvoriť jedinečný výsledok. Naliatie vody môžeme považovať za projekt len na tej najnižšej úrovni, hoci aj tak nespĺňa jednu podmienku a tou je jedinečnosť. Vodu si totiž nalievame každý deň a až na malé odchýlky je výsledok stále rovnaký.

Testovanie je veľmi úzko spojené aj s kvalitou výsledného produktu. Podľa [3] kvalita znamená prispôsobenie sa špecifickým požiadavkám. Na zabezpečenie potrebnej kvality však musí aj niekto dohliadať. Je to manažment kvality softvéru, ktorý obsahuje niekoľko základných procesov. Podľa normy IEEE12207 [5] sú to nasledovné procesy:

- Procesy na zabezpečenie kvality
- Procesy verifikácie
- Procesy validácie
- Procesy revízie
- Procesy auditov

Štandard IEEE12207 [5] definuje zabezpečenie kvality ako proces, ktorý zabezpečuje, že softvérové produkty spĺňajú špecifické požiadavky a dodržiavajú stanovené plány. Keďže tento proces beží počas celého vývoja softvérového produktu, vykonávanie jeho aktivít odhaľuje problémy, už pri ich zárodku.

Kvalitu nemôžeme testovať priamo ale je niekoľko faktorov, ktoré ju môžu spraviť viditeľnou. Kvalita má tri druhy faktorov. Funkcionalita, spracovanie a adaptabilita. Každý z uvedených faktorov sa dá ďalej rozložiť na menšie časti, ktoré sú zobrazené v nasledujúcej tabuľke [4].

Tab.1. Typické faktory softvérovej kvality

Funkcionalita (vonkajšia kvalita)	*Spracovanie (vnútorná kvalita)*	*Adaptabilita (kvalita pre budúcnosť)*
Správnosť	Efektivita	Flexibilita
Spoľahlivosť	Testovateľnosť	Znovupoužiteľnosť
Použiteľnosť	Dokumentácia	Udržiavateľnosť
Integrita	Štruktúra	

Prečo mám testovať?

Odpoveď na otázku je veľmi jednoduchá ale mnohí ju neakceptujú alebo nechcú akceptovať. Odpoveďou je napríklad aj môj citát. „Testovať je potrebné len tie veci, ktoré chceme aby fungovali.“ Teda pokiaľ si chcem byť istý funkčnou stránkou vytvoreného diela, musím si všetko overiť inak nemôžem mať istotu.

Testovanie býva spravidla naplánované na koniec životného cyklu projektu. Dôvod je prostý, jednoducho pokiaľ nie je nič naprogramované nemám čo testovať. Toto je síce pravda, ale dôležitou vecou, ktorú si musím uvedomiť je, že projekty nebývajú vždy jednoduché a teda obsahujú viac podproblémov. Testovanie všetkých týchto podproblémov až po úplnom naprogramovaní je extrémne náročné a v niektorých prípadoch pre človeka až nemožné. Preto je dôležité si testovanie naplánovať už do obdobia programovania a rozumne ich skombinovať. Mojou ideou a určite budú so mnou súhlasiť viacerí, je zahájiť testovanie najrozumnejšie po naprogramovaní každého menšieho ale samostatne funkčného celku. Tým docielim overenie funkčnosti už v priebehu programovania a vyvarujem sa mnohým rizikám. Testovanie však aj napriek tomu by malo pozostávať z troch hlavných častí:

- Testovanie komponentov – overujú sa jednotlivé fungujúce a testovateľné komponenty a skúma sa ich implementácia na základe návrhu.
- Testovanie integrity – je overovanie integrity a komunikácie medzi jednotlivými komponentmi. Odhaľuje problémy, ktoré mohli vzniknúť pri spájaní jednotlivých komponentov do komplexného systému.
- Testovanie systému – testuje sa kompletný systém ako celok a overuje sa jeho zhoda so špecifikovanými požiadavkami.

Ako a čo teda mám testovať

Prvým rizikom je už spomínané testovanie po úplnom naprogramovaní produktu. V prípade, že takéto testovanie nepreukáže žiadne chyby je samozrejme aj takýto spôsob možný. Lenže podľa môjho názoru je dosiahnutie takéhoto scenára len utópiou a realita je úplne iná.

V reálnom svete by som však pri testovaní kompletného (a ešte vôbec netestovaného!) projektu našiel takmer s istotou mnoho chýb. Čo ale s nimi. Samozrejme je ich potrebné odstrániť. Lenže oprava vôbec nie je jednoduchá. Prečo? Pretože odstránenie jednej chyby pri zle naprogramovaných produktoch s veľkou pravdepodobnosťou bude mať za následok vytvorenie ďalšej, odstránenie tej ďalšej vyvolá chybu opäť na inom mieste a dá sa očakávať, že to bude pokračovať takto ďalej. Ani nejdem spomínať náklady, či už finančné alebo časové, ktoré je nutné vynaložiť na takúto opravu. Bohužiaľ tadiaľto cesta nevedie. Okrem toho, že opravenie takto nájdených chýb je takmer nemožné, je tu aj ďalší problém a to ten, že nájdenie mnohých chýb nám určite unikne.

Testovanie po častiach odstraňuje riziká kompletného testovania alebo ich minimálne vo veľkej miere redukuje. Otestovanie menšej časti umožňuje oveľa dôkladnejšie pretestovanie všetkých funkcií a čo je najdôležitejšie, odstraňovanie nájdených chýb je neporovnateľne jednoduchšie. Pri správnom programovaní totiž očakávam, že chyba nájdená v práve testovanom celku a opravená nemá negatívny vplyv na ostané celky, teda že neovplyvní žiadny iný výpočet.

Každé testovanie však podstupuje isté riziká a dokonca aj keď sa postupuje, obrazne povedané, podľa príručky. Prvým a veľkým rizikom je zákazník. Zákazník si totiž veľmi jednoducho môže povedať, že to čo vytvorený produkt robí, nie je to čo požadoval. Áno, ja viem, že takto boli zadané požiadavky, viem, že pri špecifikácii so všetkým súhlasil, ale... Sme ľudia a nie všetci sa rozumieme tomu, čo robia iní. Aby si o nás profesionáli nemysleli, že sa tomu nerozumieme (hoci sa naozaj nerozumieme), častokrát súhlasíme aj s vecami, ktoré nám nie sú jasné. Toto sa veľmi jednoducho stane aj pri analýze a špecifikácii softvéru a tým pádom všetko čo sa na základe špecifikácie vytvára je niečo iné, ako zákazník požadoval.

Čo som týmto chcel vlastne povedať? Asi to, že testovanie je najúčinnejšie, pokiaľ je všetko konzultované aj so zákazníkom. Ja ako tester si môžem povedať, že to čo som otestoval spĺňa špecifikáciu, ale čo keď mi zákazník povie, že to nie je to, čo chcel on. Musím opraviť niečo, čo nie je z môjho pohľadu chyba alebo, ak mám dosť odvahy, povedať zákazníkovi, že je to to, čo požadoval v špecifikácii. Osobne by som ako zákazník možno súhlasil ale v budúcnosti by som tohto vývojára na podobnú prácu určite nezavolal. Takže si položme otázku. Stojí mi to za to?

Ako už len ty toto môžeš vedieť

Veľmi jednoducho. Za svoj krátky programátorský život som tiež už všeličo zažil a hoci práca pre reálneho zamestnávateľa bola naozaj len krátka, postarala sa o zážitok, ktorý je takmer určený do tejto eseje.

Pracoval som na jednom veľmi malom projekte, na ktorom som si prešiel vlastne celý životný cyklus, pretože som na ňom pracoval sám pod vedením svojho vedúceho. Vyvíjal

som softvér pre interné použitie vo danej firme, čiže „zákazníkom" bol priamo môj vedúci. Pri analýze a špecifikácii problému sme sa samozrejme na všetkom zhodli a ja som sa veľkou vervou pustil do samotného programovania. Postupoval som systémom naprogramovania jedného menšieho celku, jeho otestovanie a následná práca na ďalších celkoch s podobným scenárom. Raz týždenne som musel vedúcemu ukázať pokrok v mojej práci. Keďže všetko bolo podľa špecifikácie otestované, vedúci sa ani neobťažoval program zvlášť skúšať. Ja ako začínajúci zamestnanec som v domnienke, že asi vie čo robí, postupne pokračoval vo vývoji až do úspešného konca. Hmm. Úspešného. Nie tak celkom. Pri finálnom odovzdávaní si vedúci konečne spustil aj môj výrobok a samozrejme sa stala vec, ktorú si žiadny z programátorov a vývojárov neželá. Produkt nespĺňa požiadavky.

Ako je to možné? Veď je vytvorený presne podľa špecifikácie. Svojim manažérskym spôsobom mi vysvetlil, že ja som ten čo spravil chybu a verdikt znel prerobiť. Ďalší čas strávený nad vecami, ktoré už raz boli riešené. Po istom čase sa podarilo upraviť všetko podľa nových požiadaviek, ale keďže času bolo veľmi málo, testovanie bolo neustále odkladané na neskôr. Po ukončení programátorskej práce som vedúceho upovedomil o danej skutočnosti. Ten obratom požadoval daný výrobok s odôvodnením, že čas určený na projekt sa naplnil a je potrebné odovzdať finálnu verziu. Áno, očakávate správne. Finálne testovanie sa bohužiaľ nekonalo. A čo sa stalo? Neviem.

Existuje jeden anglický citát neznámeho autora, ktorý sa dá voľne preložiť: „Nechystám sa byť jedným z tých, ktorí zomreli rozmýšľaním nad tým, čo ak?". Bohužiaľ stal som sa jedným z takýchto ľudí. Vedúcemu som vysvetlil, že nie je mojím cieľom vytvárať niečo, čo si nemôžem overiť, že funguje a odišiel som.

Týmto krátkym ale pravdivým príbehom som chcel priblížiť, aký je pri testovaní dôležitý kontakt so zákazníkom, hoci problém sa dal odstrániť už skôr. Beriem si za chybu nesprávny začiatok projektu, teda analýzu a hlavne špecifikáciu. Ale žiadny človek učený z neba nespadol a tak ako každý, sa musí učiť na vlastných chybách. Ja nie som výnimkou a poučil som sa.

Záver

V eseji som priblížil možné riziká a dôsledky podcenenia testovania malých projektov. Z vlastnej skúsenosti viem, že niečo také sa nevypláca a spôsobí to len problémy, ktorým sa dá jednoducho vyhnúť. Je dôležité si uvedomiť, že veľké a malé projekty sú si veľmi podobné a preto, keď sa dokážeme postaviť k veľkému projektu s rešpektom a uvedomujeme si jeho riziká, mali by sme si podobné riziká uvedomovať aj na malom projekte.

„Radšej obetujem chvíľu na testovanie, ako celý život žiť v neistote a strachu pred chybami, ktoré som neodhalil."

Použitá literatúra

1. Averson, P.: The Deming Cycle, 1998. dostupné na internete: http://www.balancedscorecard.org/TheDemingCycle/tabid/112/Default.aspx
2. Gelperin, D., Hetzel, B.: The Growth of Software Testing. CACM Vol. 6, 31 (1988).

3. Chillarege, R.: Software Testing Best Practices. Center of Software Engineering, 1999.
4. Pan, J.: Sofware Testing. Carnegie Melon university,1999.
5. Rico, D.F.: IEEE12207 Software Life Cycle, 2003.

Annotation

Do not test a small project

Testing is part of every project, whether small or large, software or even not software. It is underestimated by many people and in many cases is testing on the last place in creating a software product. When it is so important, let's ask a question. Why it has so unflattering label with the word „Irrelevant!"? In my opinion, especially small projects are pushing the testing on the sideline, resulting in a waste of time spent on the project. Do not test small project! When someone says this sentence, he is not worthy working on any project. Every small project can hide a secret place, which may be very important in its core. In this essay I will point, that the testing of any small project makes sense and I will try to get him rid of label "Irrelevant!". I also explain that testing of small projects is not to prolong the process, but in many cases it saves the whole project.