AKO DOSIAHNUŤ KVALITU?

Najväčšia chyba je, že si nie sme vedomí žiadnej chyby. (Thomas Carlyle)

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava

Abstrakt. *Neexistuje dokonalý človek. Každý má svoje nedostatky, a preto každý robí chyby. Tak ako v živote tak aj vo vývoji softvéru. Niektoré menšie nemusia mať žiaden vplyv na kvalitu produktu alebo softvéru a môžu byť zanedbané. Iné väčšie môžu mať katastrofálne následky, aké si ani nevieme alebo nechceme predstaviť. Kvalita produktu je v súčastnosti veľmi dôležitá, pretože s narastajúcim počtom firiem sa zvyšujú aj nároky zákazníka. Táto esej bude rozoberať to, ako chyby vo vývoji softvéru vznikajú, aké druhy chýb existujú, ako vzniknuté chyby nájsť čo najskôr a ako ich odstrániť ešte pred tým, než spôsobia veľké škody. V mnohých prípadoch to nemusí byť jednoduché. Najúčinnejším a veľmi známym spôsobom je testovanie, ktorému bude venovaná časť eseje.*

Kľúčové slová: *Kvalita, detekcia chýb, testovací plán, testovanie*

Chápanie kvality

Existuje množstvo definícií kvality softvéru. Každá má v určitom zmysle chápania kvality pravdu, ale žiadna nevystihuje presne to, čo kvalita naozaj je. Pýtate sa prečo? Pretože kvalita je veľmi všeobecný pojem a je veľmi ťažké vyjadriť jednou či dvoma vetami jej podstatu. Každý ju môže chápať inak a pre každého môže znamenať niečo iné. Keď idete do obchodu a chcete si kúpiť topánky, neviete hneď povedať, ktoré sú kvalitné a ktoré nie. Môžete si len povedať: „tieto sú drahé, preto asi budú kvalitné“ alebo „táto značka je dobrá, mali by byť kvalitné“. Často sa však presviedčame o tom, že opak je pravdou. Problém je v tom, že to, či sú naozaj topánky kvalitné, zistíte až po niekoľkých mesiacoch

možno až rokoch nosenia. Preto by sme kvalitu mohli definovať ako mieru uspokojenia požiadaviek zákazníka. To znamená, že ak si zákazník obuje svoje topánky po piatich rokoch nosenia a prejde sa v nich v daždi po meste bez toho, aby mu do nich natieklo a boleli ho nohy, môže si povedať, že má naozaj kvalitné topánky, v prípade ak on sám vyžaduje od topánok komfort a suché nohy. Pri sofvérovom produkte to ale nemusí byť také jednoduché a aj preto vzniklo vo vývoji softvéru odvetvie s názvom Manažment kvality.

Manažment kvality

Je súčasťou riadenia podniku. Cieľom je optimalizácia pracovných postupov alebo výrobných procesov so zohľadnením materiálových a časových zdrojov, očakávanej konečnej kvality produktu a predpokladaného ďalšieho rastu a vývoja firmy. Medzi základné nástroje manažérstva kvality patria napríklad zber dát o spokojnosti zákazníka, vyhodnocovanie výkonnosti a spoľahlivosti procesov, počtu reklamácií od zákazníkov, štatistika a iné, vedúce k zlepšovaniu podnikových komunikačných štruktúr, odborných firemných stratégií, zvýšeniu spokojnosti zákazníka, úspore zdrojov vďaka zavedeniu vhodne zvolených štandardov. Podľa [8] zahŕňa manažérstvo kvality: Plánovanie kvality, zabezpečenie kvality a riadenie kvality.

Ako merať kvalitu?

Vo vývoji softvéru je množstvo aspektov, pomocou ktorých je možné určiť kvalitu sofvéru. Nazývajú sa softvérové metriky [7]. Tieto metriky nám umožňujú získať užitočné a merateľné informácie o štruktúre softvérového produktu. Prvé metriky ako napríklad LOC (Počet riadkov kódu) boli dostupné a využívané už veľmi dávno v minulosti. Dnes má softvérový inžinier k dispozícii široký výber týchto metrík. Pomáhajú mu získať potrebný pohľad na chápanie a ohodnotenie štruktúry a kvality systému. Preto je potrebné správne sa rozhodnúť pre množinu metrík, ktorá je najvhodnejšia pre dané prostredie a dokáže nám poskytnúť komplexný pohľad na konkrétny systém [7].

Rôzne štúdie sa pokúšajú zladiť softvérové metriky s kvalitou, alebo potvrdiť či vyvrátiť význam viacerých metrík vyskytujúcich sa v literatúre. Niektorí používajú softvérové metriky na to, aby predpovedali náchylnosť sofrvéru k chybe od ranných fáz vývoja. Okrem iného sa snažia vyhodnotiť aj to, či používanie týchto metrík, má vôbec vplyv na úsilie, ktoré je potrebné vynaložiť na udžanie systému v aktuálnom stave vzhľadom k meniacim sa požiadavkám. Toto tak isto môžu byť ukazovatele kvality systému.[7] Metódy spojené so softvérovými metrikami sa často používajú aj na identifikovanie častí systému, ktoré s najväčšou pravdepodobnosťou práve kvôli chybám nefungujú správne alebo vyžadujú príliš náročnú údržbu.

Norma ISO9126 definuje tieto metriky kvality: funkčnosť, efektívnosť, použiteľnosť, prenosnosť, udržateľnosť a spoľahlivosť. Ak chceme vedieť čo jednotlivé vlastnosti znamenajú, stačí ak si pri každej z nich položíme otázku:

Funkčnosť - Sú požadované funkcie k dispozícii v našom softvéri?
Efektívnosť – Aký efektívny je náš softvér?
Použiteľnosť – Je softvér jednoduchý na použitie ?
Prenosnosť – Aké ľahké je preniesť softvér do iného prostredia?
Udržateľnosť – Aké ľahké je modifikovať softvér?
Spoľahlivosť – Aký spoľahlivý je softvér?

Spoľahlivosť

Za jednu z najdôležitejších ja osobne považujem spoľahlivosť, lebo je to pohľad na kvalitu zo strany používateľa teda nášho zákazníka. A ako sa hovorí „náš zákazník, náš pán". Túto vlastnosť môžeme tiež vnímať ako nepriamu úmernosť intenzity chýb a teda, čím menej chýb program obsahuje, tím je väčšia jeho spolahlivosť. Podľa zdroja [1] je spoľahlivosť definovaná ako pravdepodobnosť, že softvér bude pracovať bez chýb počas stanovenej „záručnej" doby. Zlyhanie procesu a teda spolahlivosť systému je závislá aj od prostredia, v ktorom je program vytváraný. Podcenenie spoľahlivosti môže v niektorých prípadoch spôsobiť vážne problémy. Napriek tomu, že produkt má vynikajúcu funkcionalitu, pracuje veľmi rýchlo a presne podľa požiadaviek zákazníka, nie je použitelný, ak často dochádza k poruchám či výpadkom. Práve toto je väčšinou následkom veľkého množstva chýb. Teda ak chceme dosiahnuť vysokú spoľahlivosť, musíme zabezpečiť, aby náš softvér obsahoval čo najmenšie množstvo takých chýb, ktoré sú pre používateľa nepriateľné.

Chyby boli, sú aj budú

Chyby sú ako baktérie. Existujú všade okolo nás a mnohokrát o nich ani nevieme. Náklady na ich odstránenie prudko stúpajú s časom. Čím neskôr chybu objavíme, tím väčšie problémy máme s jej odstránením v neskorších fázach vývoja, či už z hľadiska finančného, časového, alebo kvôli riziku, že pri jej odstraňovaní urobíme ďalšie chyby. Preto po každej fáze treba produkt dostatočne vyskúšať a skontrolovať. Je nereálne odstrániť všetky sofvérové chyby pred tým, ako bude produkt spustený do prevádzky. Myslím si, že aj keby sa to náhodou niekomu podarilo, bolo by to určite nákladovo neefektívne. Musel by na to vynaložiť oveľa viac času a úsilia, okrem iného aj kvôli skutočnosti, že by to musel testovať v prostredí, kam sa má konečný softvérový produkt nasadiť. Je omnoho jednoduchšie a efektívnejšie odstrániť drobné závady zistené v záverečných fázach testovania. Aby sme boli schopní chyby odstrániť, musíme sa najskôr zamyslieť nad tým, akými spôsobmi vlastne môžu vzniknúť. Podľa [1] môžu chyby vznikať nasledovne:

1. pri vytváraní nového kódu

2. pri pôvodnom návrhu

3. pridávaním nových funkcií

4. počas zmien návrhu

5. opravovaním nájdených chýb

Určite je aj veľa iných miest, kde môžu chyby vznikať a je len na vývojároch alebo skôr na nástrojoch, ktoré použijú, aby tieto chyby objavili. Množstvo chýb vzniká tiež vtedy, ak nejakú časť kódu zdedíme z nejakej inej aplikácie. V takýchto situáciach vzniká otázka, či je efektívnejšie nanovo napísať vlastný kód, alebo ho prevziať z už existujúcej aplikácie a vynaložiť čas na úpravu a odstránenie vzniknutých chýb tak, aby fungoval podľa našich predstáv.

Sú rôzne kritériá a schémy klasifikácie chýb podľa rôznych štadardov a je veľa spôsobov ako ich charakterizovať.

Chyby dostali podľa [6] jedno z nasledujúcich hodnotení závažnosti na základe ich potenciálneho dopadu na zlyhanie.

1. kritická: chyba, ktorá by mohla spôsobiť výpis z pamäte aplikácie, servisnú odstávku alebo reboot systému

2. majoritná: chyba, ktorá by mohla spôsobiť segmentačnú chybu, alebo zníženie výkonu ako napríklad úniky pamäte, miznutie zdrojov, nevratné poškodenie dát

3. minoritná: chyba, ktorá môže mať za následok nestále a neočakávané správanie, ale nemusí mať veľký vplyv na systém

4. programovací štandard: kód, ktorý porušuje štandard kódovania, ktorý má potenciál znižovať udržiavateľnosť a čitateľnosť softvéru

A čo s nimi ?

Ideálna infraštruktúra softvérového testovania by mala umožniť vývojárovi nájsť a odstrániť čo najviac chýb čo najskôr a s využitím čo najmenších nákladov. To je samozrejme možné len do určitej miery. A ak chceme dosiahnuť čo najvyššiu kvalitu, musíme sa len snažiť tomuto ideálu čo najviac priblížiť.

Na zvýšenie kvality softvéru je vytvorených množstvo nástrojov a je len na nás, ktorý alebo ešte lepšie ktoré si vyberieme. Pri výbere testov treba byť naozaj opatrný, aby sa nestalo, že si zvolíme také typy metód, ktoré nám zaberú množstvo času a zvýšia kvalitu len minimálne. Alebo na druhej strane nám síce nezaberú veľa času, ale ani nepomôžu nájsť dostatočné perceto chýb v programe.

Testovanie pozostáva z týchto základných úrovní:

1 integračné testy
2 systémové testy
  2.1 funkčné testy
  2.2 nefunkčné testy
3 systémové integračné testy
4 regresné testy
5 akceptačné testy

Testovanie by malo podľa [2] iniciovať vykonanie každého výroku aspoň raz a vykonať ho vo všetkých možných rozhraniach. Nie je ale možné testovať všetky kombinácie ciest. Bolo by to neefektívne ako kvôli časovým nárokom, tak kvôli nárokom na réžiu. Preto testy a vstupné dáta musia byť starostlivo vybrané a to tak, že úspešné prevedenie výslednej množiny testov bude vypovedať o správnosti maximálnej množiny funkcií. Z toho usúdime, že program je s veľkou pravdepodobnosť správny. Je samozrejmé, že každá nájdená chyba musí byť odstránená. No na druhej strane pri každom odstránení poruchy sa vytvorí nový program, ktorý tiež nemusí byť správny. Preto sa zodpovedajúci výber testov musí zopakovať, aby sme sa uistili, že nevznikli žiadne nové nedostatky. Tento spôsob je špecifický pre regresné testovanie.

Všetky zistené nedostatky by mali byť zaznamenané a analyzované. Ak zistíme, že v jednej časti vývoja je veľký výskyt chýb, môže to poukazovať na slabú spoľahlivosť a takáto fáza by mala byť nanovo navrhnutá.

Keď sú všetky úrovne testovania úspešne dokončené, je často preferované, aby samostatné kvalifikačné alebo akceptačné testy boli spustené pred tým, ako výsledný produkt vydáme zákazníkovi. Teda mali by prebiehať mimo zákazníka. Toto by malo byť pokryté vo všetkých aspektoch špecifikácie požiadaviek a ideálne by bolo spustenie v cieľovom prostredí.

Zákazník by mal byť pozvaný, aby pripravil alebo odsúhlasil podmienky skúšobnej špecifikácie a mal by byť svedkom chodu testu. Po úspešnom dokončení týchto testov môže byť produkt vydaný zákazníkovi.[2]

Plán testovania [2]

Predtým, ako budeme softvér testovať, musíme mať naskôr vypracovaný dôkladný plán testovania. Ten nám zebzpečí , že dokážeme systém testovať čo najefektívnejšie a tým zaručíme vysokú spolahlivosť a minimalizujeme počet chýb.

Plán testovania systému navrhuje techniky pre detekciu a odstránenie chýb v každom štádiu vývoja softvéru. Plán môže byť aktualizovaný a dolaďovaný počas celého vývoja, ak sú dostupné ďalšie informácie. To zahŕňa prípady, kedy sú pridané nové položky na testovanie, napríklad pri zmene návrhu alebo požiadaviek. Toto rozšírenie testovacej množiny smeruje k detailnejšiemu návrhu a implementácii. Testovací plán

projektu je určený na to, aby poskytol základné informácie pre identifikáciu objektov nášho testovania, informácie o čase testovania, spôsobe testovania, o tom, ako posudzovať výsledky a na koniec, aké dôsledky z toho všetkého všetkého vyvodiť. V testovacom pláne by mali byť zahrnuté nasledujúce sekcie:

Úvod

Je to súhrn softvérových položiek a funkcií, ktoré sa majú testovať. Úvod by mal naznačovať celkový prístup k testovaniu, ktorý bude použitý v celom projekte. Môže tiež hovoriť o rôznych obmedzeniach, ktoré majú konkrétny vplyv na testovanie projektu.
To poskytuje:

1. prehľad o štruktúre
   - aké sekcie bude obsahovať, rozhodnutie o tom, či to bude jeden dokument alebo súbor súvisiacich dokumentov
2. opis prístupu k testovaniu
   - metódy a nástroje, ktoré majú byť použité počas projektu a pre aké presné prístupy k testovaniu v rôznych častiach sme sa rozhodli
3. správa o prípadných obmedzeniach
4. vyhlásenie o tom či má byť test aktualizovaný počas projektu

Testované položky

Táto sekcia by mala identifikovať všetky testované položky, alebo poskytnúť prostriedky na ich rozpoznanie, ak sa vyskytnú počas vývoja. Identifikácia by mala obsahovať verziu alebo stupeň revízie.

Testované funkcie

Táto sekcia sa snaží identifikovať všetky softvérové funkcie a súbory funkcií, ktoré sa majú testovať. Uvádza tiež metódy a s nimi súvisiace testy, akými sa dané funkcie a súbory funkcii budú testovať.

Netestované funkcie

Tu budú uvedené všetky funkcie a súbory funkcií, ktoré sa testovať nebudú spolu s odôvodneniami, prečo ich netestovať.

Nefunkcionálne testovanie

Zahŕňa všetky nefunkcionálne testovania ako napríklad, ako sa bude správať systém pod tlakom, či sú bezpečnostné opatrenia dostatočné, alebo či sú požiadavky, ako je napríklad doba čakania medzi poruchami uspokojené. Čo bude akceptačné a implementačné testovanie vyžadovať?

Prístup k testovaniu

Ukazuje základný prehľad častí produktu, ktoré budú testované a testovaciu stratégiu, ktorá bude použitá na zabezpečenie všetkých požiadaviek. Stratégie a obmedzenia, ktoré sú uvedené v pláne kvality, sú založené aj na základných aj na špecifických požiadavkách ako sú oprávnenia, špecifikácie a zmluvy. Testovací plán poskytuje okrem iného aj

podrobnejší opis testovacích aktivít, techník a nástrojov. Táto sekcia popisuje hlavné činnosti, techniky a nástroje, ktoré sú používané na testovanie jednotlivých kombinácií alebo množín funkcií. Testovací prístup musí byť opísaný podrobne, aby umožnil identifikáciu hlavných testovacích úloh a odhad času pre každú z nich.

Obmedzenia ovplyvňujúce prístup k testovaniu

Táto časť identifikuje obmedzenia, ktoré budú mať vplyv na prístup k testovaniu.

1. dostupnosť položiek, ktoré sú testované
2. časové obmedzenia
3. finančné prostriedky
4. počet voľných pracovníkov
5. praktické skúsenosti pracovníkov
6. hardvérové prostriedky
7. dostupnosť stroja
8. dostupnosť podstatných hardvérových a softvérových prostriedkov
9. dizajn výrobku
10. vzťah so zákazníkom
11. typ výrobku
12. ťažkosti pri predpovedaní výstupu

Ako je prístup k testovaniu ovplyvnený obmedzeniami

Táto sekcia identifikuje, ako čo najefektívnejšie použiť zdroje dostupné pre testovanie. Je tu tiež uvedené, ktorá stratégia bola použitá a prečo bola zvolená.

Organizácia testovacích aktivít

Poskytuje opis toho, ako sa bude testovanie vykonávať.

1. optimálne využitie dostupných zdrojov
2. prekonanie všetkých problémov súvisiacich so splnením veľkých časovo náročných testov

V testovacom pláne môžu byť zahrnuté aj tieto sekcie, ktoré nebudem podrobnejšie charakterizovať ako napríklad: Plánovanie testovacej činnosti, pohotovostné plány, personálne a vzdelávacie potreby, vyhovel / nevyhovel kritériám pre body v rámci testu, testovacie povinnosti, testovacie úlohy, schopnosť dodania testu, potreby prostredia, postupy a zariadenia pre aktualizáciu testovacieho plánu počas celej doby trvania projektu a iné.

V testovacom pláne je naozaj množstvo sekcií, ktoré treba zanalyzovať skôr ako začneme s testovaním. V rôznych firmách môžu byť používané rôzne plány testovania. Ale nemalo by sa stať, že firma nemá žiadny plán. Testovací plán by mal poskytnúť akési médium pre komunikáciu medzi návrhárom výrobku a návrhárom testov, tak aby podrobnosti o návrhu výrobku skôr pomáhali ako bránili testovaniu výrobku.

Záver

Veľakrát sa stáva, že firmy používajú množstvo nástrojov na zvýšenie kvality a myslia si, že tým automaticky zlepšujú aj kvalitu produkovaného systému. Známy a užitočný nástroj používaný mnohými firmami je refactoring. Zdroj [7] hovorí o tom, že aj refactoring nie vždy zvýši kvalitu softvéru. Tento spôsob môže zvýšiť jeden aspekt kvality, ako je udržateľnosť, ale je otázne či má vplyv aj na iné ukazovatele kvality. Toto by sme mohli povedať aj o iných podobných nástrojoch. Preto vývojár musí vedieť, kde a ako ich používať, aby daný nástroj naozaj robil to, na čo je určený a tým prispieval k zvýšeniu kvality softvéru.

Takže na záver si znovu položme otázku: Ako vlastne dosiahnuť kvalitu? Odpoveď je veľmi nejednoznačná. Kvalita je veľmi relatívny pojem, a preto ju podľa môjho názoru, ani nie je možné v plnej miere dosiahnuť. Veď vždy sa nájde niekto, kto daný softvérový produkt za kvalitný považovať bude a niekto, kto ho bude zatracovať. Každý má na kvalitu iné nároky a tak si myslím, že je možné iba určiť, do akej miery sme splnili niektoré konkrétne nároky či požiadavky a do akej miery sme dosiahli rôzne metriky softvérovej kvality. Napĺňať tieto požiadavky je možné rôznymi spôsobmi. Ja, keďže mám rád vo veciach systém a istú hierarchiu a poriadok, považujem za najvhodnejšie vypracovať podrobný testovací plán, ktorým sa neskôr pri testovaní riadime. Tento plán by mal určovať systematické a efektívne metódy testovania, ktoré by boli starostlivo vyberané a pre každý konkrétny softvérový produkt alebo fázu vývoja špecifické. Toto bremeno výberu leží len a len na vývojárovi . To, ako sa s touto úlohou popasuje, sa určite odzrkadlí aj na kvalite softvéru, ktorý odíde z jeho dielne.

Použitá literatúra

1. Tivari,A. Tandon,A.: SHAPING SOFTWARE QUALITY - THE QUANTITATIVE WAY, Pages 84 – 94, Publication Year 1994
2. G. D. Frewin, B. J. Hatton: Quality Mnagement – procedures and practices,

Pages: 29 – 38, 1986

3. Manthos A. Tsoukarellas, Vasilis C. Gerogiannis, Kostis D. Economides: Systematically Testing a Real-Time Operating system, 1995
4. Gautham Reddy N.: Testing processes in business-critical chain software lifecycle,

Tata Consultancy Services Ltd, Boston, MA, USA 2009

5. Gregory Tassey: The Economic Impacts of Inadequate Infrastructure for Software Testing, National Institute of Standards and Technology, May 2002
6. Jiang Zheng, Laurie Williams, Nachiappan Nagappan, Will Snipes, John Hudepohl, Mladen Vouk: A Study of Static Analysis for Fault Detection in Software, 2006

7. Konstantinos Stroggylos, Diomidis Spinellis: Refactoring – Does it improve software quality? Fifth International Workshop on Software Quality © 2007
8. William R. Duuncan: A guide to the project management body of knowledge,
Project management institute, 1996

Annotation

How to reach quality?

There is no perfect man. Each has its shortcomings, and therefore everyone makes mistakes. As in life, both in software development. Some minor may not have any impact on product quality or software and can be neglected. Others may have more dire consequences, what you dont know or do not want to imagine. Product quality is very important at present because of the increasing number of companies is also increasing customer requirements. This essay will discuss how defects in software development created, what types of errors exist, how to find the error as soon as possible and remove them even before they cause major damage. In many cases, it might not be easy. The most effective and widely-known method is testing, which will be devoted to the essay section.