CHCEME KVALITNÝ SOFTVÉR!

Rozdávaj radosť okolo seba a svet bude krajší!

Michal Masliš

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
majki777[zavináč]gmail[.]com

Abstrakt. *V súčasnosti na trhu pôsobí mnoho firiem, ktoré vytvárajú veľké množstvo rôzneho softvéru. Každá firma sa snaží vytvoriť taký softvér, s ktorým by bol zákazník spokojný. Zákazníci chcú kvalitný softvér, no často sa však stáva, že s kvalitou zákazníci až tak spokojní nie sú. Táto nespokojnosť so sebou prináša aj stratu drahocenného času, nové investície, stratu zisku. V eseji sa uvažuje nad tým, ako sa dá táto nespokojnosť zákazníkov minimalizovať. Je tu niekoľko rád, ako postupovať pri tvorbe kvalitného softvéru, ako ho testovať. V eseji sa uvažuje taktiež nad dôležitosťou kvality softvéru a efektívneho testovania, ktoré by odhalilo čo najviac chýb a taktiež by preukázalo požadované vlastnosti softvéru.*

Kľúčové slová: *kvalita, testovanie*

Úvod

V 19. storočí bol svet úplne iný ako je dnes. Svet nemal toľko technických prostriedkov ako má dnes. V súčasnosti je na Slovensku takmer v každej domácnosti počítač, počet aktívnych SIM kariet je vyšší ako počet obyvateľov. Každý počítač alebo mobilný telefón potrebuje na svoju činnosť softvér, bez neho by bol iba „kus železa“. Tento softvér musí niekto vytvoriť a ten niekto je človek. Keďže človek nie je neomylný, často sa stáva, že aj človekom vytvorený softvér sa niekedy nespráva tak, ako by sa očakávalo.

Softvér môže vyvíjať jednotlivec, ale taktiež aj skupina ľudí, ktorí pracujú v tíme, napríklad v nejakej firme. Aby firma mohla v konkurenčnom prostredí prosperovať, potrebuje zákazníkov. Zákazníci však nie sú spokojní s firmou, ktorá im vytvorí nekvalitný softvér. Ak zákazníci nie sú spokojní a majú k dispozícii iné lepšie alternatívy, v budúcnosti služby takejto firmy pravdepodobne využívať nebudú. Ak si chce firma udržať a získať nových zákazníkov, potrebuje vytvárať softvér, s ktorým budú zákazníci

spokojní. Potrebuje vytvárať kvalitný softvér, ideálne bez chýb. Existujú však nejaké postupy ako vytvoriť kvalitný softvér? Ako ho testovať?

Čo je to vlastne kvalitný softvér?

Existuje viacero noriem, ktoré sa zaoberajú kvalitou softvéru. Norma ISO 9126 hovorí o šiestich vlastnostiach, ktoré by mal mať každý kvalitný softvér: funkčnosť, spoľahlivosť, použiteľnosť, efektivita, udržovateľnosť a prenositeľnosť [2].

Kvalita je podľa ISO 8402 súhrn vlastností a charakteristík výrobku, procesu alebo služby, ktoré preukazujú jeho schopnosť splniť určené alebo odvodené potreby. Kvalita je vlastne stupňom splnenia požiadaviek alebo potrieb. Ako pani prof. Bieliková vraví: „Ak cieľom bude vyvinúť nefunkčný softvér, potom čím menej bude fungovať, tým bude kvalitnejší“ [3].

Definovanie požiadaviek je veľmi dôležité, pretože ak sa tieto požiadavky zle definujú, výsledkom bude to, že pri tvorbe kvalitného softvéru sa budeme snažiť tieto zle definované požiadavky splniť, no zákazník s tým nebude spokojný, pretože očakával niečo iné. Takto vzniknú zbytočné problémy, ktorým sa dalo vyhnúť správnym definovaním požiadaviek.

Existuje veľa nástrojov na zabezpečenie a dosiahnutie kvality, jedným z nich je napríklad systém riadenia kvality podľa normy ISO 9001.

Ak chceme vytvoriť kvalitný softvér, musíme na to myslieť už od začiatku tvorby softvéru a podľa zvolenej taktiky postupovať pri tvorbe softvéru. Neexistuje spôsob, ktorý nám s istotou zaručí kvalitný softvér alebo ktorý nám zaručí softvér bez chýb. No existujú postupy, metódy, s ktorými by náš softvér mal byť kvalitnejší a mal by obsahovať menej chýb.

Reštrukturalizácia (angl. refactoring)

Reštrukturalizácia je definovaná ako proces zmeny vnútornej štruktúry existujúceho kódu pri zachovaní integrity funkčných požiadaviek kódu [4]. Reštrukturalizácia je veľmi dobrá vec. Vďaka nej sa napríklad výrazne zlepší čitateľnosť kódu, zjednoduší sa jeho štruktúra, zlepší sa jeho udržateľnosť, výkonnosť.

Za najdôležitejšiu vec pri reštrukturalizácii osobne pokladám správne pomenovanie premenných. Začínajúci programátori často totiž pomenúvajú premenné tak, že keď si prečítame ich názov, no nič nám to nehovorí (napr. a,aa,aaa, prem, prem2). Takýto spôsob nie je veľmi dobrý, pretože ak si autor po čase prečíta svoj zdrojový kód, ťažko sa mu bude v ňom orientovať, veľa času mu zaberie pochopenie, ktorá premenná čo vyjadruje. Ak si tento kód bude čítať niekto iný, bude sa mu v ňom orientovať ešte ťažšie.

Názvy premenných, tried, metód, parametrov by sa mali pomenúvať tak, aby z názvu bolo jasné, o čo ide (int counter, int getSize(),Class Animal,…). V prípade, že treba v kóde opraviť nejaké chyby alebo doplniť funkcionalitu, tak v takomto kóde sa oveľa jednoduchšie orientuje, netreba písať komentáre, ktoré by vysvetľovali, aká premenná čo znamená.

Z vlastnej skúsenosti odporúčam nepomenúvať premenné v slovenčine, ale pomenúvať premenné v angličtine, pretože takémuto kódu bude rozumieť každý programátor. Ak sa vyskytne nejaký problém, často hľadáme pomoc na internete, vtedy

nám môže poradiť hocikto. Ak by bol kód v slovenčine, programátori, ktorí nehovoria po slovensky, by mu dosť ťažko rozumeli.

Je potrebná dokumentácia?

Mnoho programátorov má pocit, že písať dokumentáciu je zbytočné. No ja si myslím, že celkom zbytočné to nie je.

Predstavme si firmu, ktorá zamestnáva človeka, ktorý sám vytvoril a spravuje nejaký zložitý systém a celé fungovanie firmy stojí na tomto systéme. Ak k tomuto systému neexistuje žiadna dokumentácia, celá firma je na tomto človeku závislá. Takýto človek potom môže napríklad „vydierať" firmu – ak mu nezvýšia plat, odíde. Keďže firma tohto človeka potrebuje a nie je nikto iný, kto by ho bez veľkých nákladov nahradil, bude mu musieť vyhovieť. Ak by bola dostupná dokumentácia, „s vydieraním" by takýto programátor veľmi nepochodil.

Iným príkladom môže byť programátor, ktorý vytvoril softvér a medzičasom odišiel z firmy. Po čase sa zistí, že sa softvér nespráva tak, ako by sa mal správať. Ak nie je k dispozícii dokumentácia, nájdenie chyby nebude až také jednoduché ako by to bolo s dokumentáciou.

Dokumentácia je teda dobrá na to, aby aj iní ľudia ako samotný tvorca softvéru mohli ľahko pochopiť, ako softvér funguje. S dokumentáciou budú vedieť omnoho jednoduchšie takýto softvér modifikovať. Dokumentácia sa okrem uvedených príkladov často využíva napríklad aj vo voľne šíriteľných softvéroch či pri testovaní. Myslím si, že v súčasnosti je už dokumentácia neodmysliteľnou súčasťou softvéru.

Čo ešte môže vplývať na kvalitu?

Na svete neexistujú dvaja ľudia, ktorí by boli dokonale identický a mali by úplne rovnaké vedomosti a zmýšľanie. Takisto je to aj u programátorov. Rôzni programátori majú rôzne schopnosti, nehodia sa všetci na všetko. Preto je dôležité priradiť správne úlohy správnym ľuďom. Aj výborní programátori sú len ľudia, preto sa niekedy môže stať, že v kritických situáciách programátori nebudú podávať dobré výkony. Aby sa chce firma snažiť takýmto situáciám vyhnúť, mala by si dávať pozor na zbytočné preťažovanie programátorov. Vo firme by podľa mňa mala vládnuť pokojná a radostná atmosféra, aby sa tam všetci dobre cítili. Som presvedčený, že toto má veľký vplyv na zamestnancov a na ich výkon.

Testovanie

Testovanie je veľmi dôležitou fázou pri vytváraní softvéru. Testovanie dokáže odhaliť mnoho chýb. Samozrejme neexistuje spôsob, ktorým, by sa odhalili všetky chyby. Preto aj po rozsiahlom a prepracovanom testovaní sa stáva, že sa vyskytnú nové chyby. No ich počet je po ukončení testovacej fázy často oveľa menší ako pri softvéri, ktorý sa netestoval. Existujú rôzne druhy, metódy, postupy pri testovaní, pokúsim sa opísať aspoň niektoré z nich.

Statické testovanie

Cieľom statického testovania je nájsť v programe preklepy, logické chyby, chyby pri práci so vstupom alebo s výstupom. Statické testovanie je najjednoduchšou a najlacnejšou formou testovania. Pri statickom testovaní sa kontroluje zdrojový kód programu čítaním a hľadajú sa v ňom chyby. Takéto testovanie môže vykonávať buď samotný programátor, ktorý tento zdrojový kód napísal, alebo to môže byť niekto úplne iný - nezainteresovaná osoba. Ak toto testovanie robí nezainteresovaná osoba, táto osoba číta zdrojový kód, ktorý predtým nevidela, preto jej trvá dlhšie, kým ho pochopí. Avšak takáto osoba skôr odhalí chyby, na ktoré programátor nemyslel. Taktiež skôr odhalí nepochopenie požiadaviek programátora. Podľa môjho názoru by mal každý programátor na svojom vlastnom kóde vykonávať statické testovanie. Ak je k dispozícii dostatok času, mohla by toto testovanie vykonávať aj nezainteresovaná osoba, ktorej výhoda je tá, že skôr odhalí chyby, na ktoré programátor nemyslel.

Dynamické testovanie

Dynamické testovanie je ďalšia forma testovania. Pri dynamickom testovaní sa testuje program po spustení. Zadávajú sa vstupy a kontrolujú sa výstupy. Toto testovanie môže vykonávať tester alebo skupina testerov. V niektorých prípadoch je však výhodnejšie na takéto testovanie použiť nejaký skript. S dynamickým testovaním pomocou skriptu sa stretávajú napríklad aj študenti našej fakulty na predmete Operačné systémy, kde sa na teste pomocou príkazu otestuj hneď dozvedia, či ich program má požadované výstupy. Neviem si predstaviť, ako by toto dynamické testovanie na tomto predmete vyzeralo, keby ho mala vykonávať skupina testerov. Asi by sa to ani nedalo zrealizovať.

Často sa však využíva kombinovaný spôsob testovania. Tester uloží operácie do skriptu, skript následne kontroluje výstupy programu automaticky. Automaticky sa však nedá skontrolovať všetko, preto po fáze automatického testovania tester môže skontrolovať správanie sa programu aj manuálne zadávaním vstupov a sledovaním výstupov.

Vývoj riadený testovaním

Tento spôsob testovania sa nevykonáva až po vytvorení softvéru, ale sprevádza tvorbu softvéru už od jeho začiatku. Toto testovanie vykonáva programátor, ktorý si ešte pred vytvorením modulu vytvorí testy pre tento vytváraný modul. Keďže modul nie je vytvorený, po spustení testu by mali byť výsledky všetkých testov negatívne. Programátor sa po vytvorení týchto testov snaží vytvoriť modul, ktorý by prešiel všetkými testami. Ak sa mu to podarí, v ďalšej fáze sa snaží vytvorený zdrojový kód zefektívniť a optimalizovať. Jednotlivé kroky pri tomto testovaní podľa [1] teda sú:

1. Napísanie malého počtu automatických testov
2. Spustenie týchto testov – uistenie sa, že pri prázdnom kóde tieto testy dávajú negatívne výsledky
3. Implementovanie kódu tak, aby prešiel testom
4. Opätovné spúšťanie testov pri pridávaní a úprave kódu dovtedy, kým všetky testy nebudú úspešné

5. Reštrukturalizácia(Refactoring) kódu alebo testovanie kódu, aby sme sa uistili, že reštrukturalizácia kódu nespôsobila negatívny výsledok nejakého testu
6. Periodicky (najlepšie raz alebo viackrát denne) spúšťanie testov pre všetky moduly, aby sme sa uistili, že správanie sa jedného modulu negatívne neovplyvňuje správanie sa iného modulu.

Tento spôsob testovania má podľa mňa najmä tú výhodu, že programátor hneď vie, či ním vytvorený kód naozaj funguje správne a či neovplyvnil správanie sa celého systému. Takýto spôsob testovania taktiež núti programátorov ešte pred vytvorením softvéru poriadne sa zamyslieť nad jeho štruktúrou. Tento spôsob je dobrý aj kvôli tomu, že v prípade chyby sa chyba hneď odhalí. Samozrejme že sa odhalia iba tie chyby, ktoré sledujú testy. Nikdy nemôžeme s istotou povedať, že sa odhalili všetky chyby.

Nevýhodou tohto testovania je podľa mňa najmä to, že programátor musí stále písať testy pre vytvárané moduly, čo si vyžaduje veľa času a oneskoruje to celkový čas vytvorenia softvéru. Ďalšou nevýhodou je to, že ak programátor vytvorí zlý test, ním vytvorené moduly môžu obsahovať chyby.

Myslím si, že toto testovanie nie je až tak vhodné pre programátorov, ktorí sú expertmi. Takíto programátori podľa mňa nezvyknú robiť chyby často, pri vytváraní jednoduchých modulov by ich podľa mňa vytváranie testov iba zbytočne zdržiavalo. Tento spôsob je podľa mňa vhodnejší pre programátorov, ktorí vedia programovať, ale nemajú až tak veľa skúseností a častejšie robia chyby.

Záver

Každý, kto vytvára nejaký softvér, by sa mal snažiť o to, aby tento softvér bol čo najkvalitnejší. V eseji som navrhol a uvažoval nad niekoľkými radami, ako postupovať pri tvorbe kvalitnejšieho softvéru, ako ho testovať. Softvér vždy bude mať nejaké chyby, no ak sa tvorca softvéru viac sústredí na kvalitu a na testovanie, je pravdepodobné, že zákazník bude so softvérom spokojnejší a že softvér bude obsahovať menej chýb. Je už na každom z nás, akú taktiku si zvolíme pri vytváraní softvéru. Dosť bolo čítania, poďme teda „kódiť kvalitný softvér"!

Použitá literatúra

1. Smith, S., Stoecklin, S., Serino, C.: *An innovative approach to teaching refactoring: Technical Symposium on Computer Science Education: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, str. 349-353, Houston, Texas, USA, 2006, Dostupné na internete: http://portal.acm.org/citation.cfm?id=1121341.1121451 [cit: 2009 október]
2. Bhat, T., Nagappan N.: *Evaluating the efficacy of test-driven development: industrial case studies: International Symposium on Empirical Software Engineering: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, str. 356-363, Rio de Janeiro, Brazil, 2006, Dostupné na internete: http://portal.acm.org/citation.cfm?id=1159733.1159787 [cit: 2009 október]

3. Bhatti, S. Z.: *Why quality?: ISO 9126 software quality metrics (Functionality) support by UML suite: ACM SIGSOFT Software Engineering Notes*, str. 1-5, 2005, Dostupné na internete: http://portal.acm.org/citation.cfm?id=1050849.1050860 [cit: 2009 október]
4. Bieliková, M.: *Úvod do manažmentu v softvérovom inžinierstve*, Dostupné na internete: http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/prednasky/msi01.pdf [cit: 2009 október]

Annotation

We want quality software!

Nowadays there are a lot of companies in the world, which create a lot of software. Every company tries to create software, that is customer satisfied with. Customers want quality software, but in many cases happends, customers are not satisfied with quality of the software. That brings some negative aspects like loss of time, new investments, decrease of profit. This essay deals with software quality. There are suggested some tips in this essay, how to proceed by creating quality software and how to test software.