Kvalita softvérových produktov a jej overovanie

BC. ĽUBOMÍR VARGA

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

Abstrakt. Vo svete softvéru je niekoľko častí a odvetví, ktoré majú za úlohu zabezpečiť jeho kvalitu. Kvalita softvéru je ťažšie merateľná ako kvalita iných výrobkov. Pri softvéri existuje viacero pohľadov na kvalitu, z ktorých sú pre rôznych hráčov dôležité rôzne pohľady. Samotný vývoj softvéru však tiež nie je unifikovaný proces. Môže prebiehať rôznymi smermi, ktoré majú diametrálne odlišné postupy vývoja i ciele. Zabezpečenie kvality je možné dosiahnuť napríklad i za pomoci starostlivého testovania, čo sa v tejto práci snažím analyzovať. Predostriem čitateľovi môj názor na statické i dynamické spôsoby testovania, priblížim mu výhody a riziká rôznych postupov, klady a zápory automatizácie testovania.

Úvod

Kvalita. Vcelku všedné slovo, avšak nájsť exaktnú definíciu tohto slova nie je možné. Väčšinou je táto vlastnosť vecí definovaná ako úroveň spokojnosti používateľa, vlastníka danej veci s používaním alebo vlastníctvom. Niekde je kvalita definovaná ako výdrž pri štandardnom používaní, poprípade kvalitu výrobku určuje i počet odjazdených kilometrov bez poruchy. Definícií je viacero, ale ktorá sa aplikuje na softvérové produkty najlepšie?

V tejto eseji rozoberám kvalitu ako atribút softvéru, ktorý sa dá merať a ovplyvniť. Esej by nemala byť zdrojom kompletného prehľadu v oblasti zabezpečenia kvality pri procese tvorby softvéru, ale iba akýmsi ľahkým úvodom do témy, ktorý naznačí pojmy a popíše časť problémovej oblasti. V oblasti, na ktorú som sa zameral, rozoberám problematiku testov a ich vykonávania do väčšej hĺbky, avšak odporúčam pre ucelený pohľad na vec preštudovať ešte aspoň nejakú literatúru vzťahujúcu sa k problematike.

Môj aktuálny názor prezentovaný v tejto eseji je taký, aký je, avšak je možné, že po preštudovaní oblasti ešte hlbšie sa v budúcnosti zmení, tak ako sa menil doteraz spoznávaním nových a nových techník testovania a nových podporných nástrojov.

Manažment v softvérovom inžinierstve, október 2007, s. 1-6.

Kvalita?

Kvalita je často chápaná rôzne. Zákazník si o výrobku pomyslí, že je kvalitný, keď mu dlho a bezproblémovo, prípadne efektívne slúži. Dnes všeobecne kvalita výrobkov upadá, keďže je z pohľadu výrobcov neefektívne vyrábať trvácne výrobky. Trvácne (kvalitné) výrobky nie je potrebné po zakúpení opravovať, udržiavať a i ich výmena nie je nutná v blízkom horizonte času. Na trhu sa ale nájdu i firmy a spoločnosti, ktorých meno je založené na kvalite. Tieto firmy produkujú výrobky nadpriemernej akosti. Tým si udržiavajú priazeň zákazníkov, ktorí sú následne ochotní za kvalitu zaplatiť. Teda pridaná hodnota výrobkov je v povestnej kvalite a zákazník, spotrebiteľ, si v prípade, že chce kvalitu, priplatí. V softvéri sa však kvalita berie jemne odlišným meradlom. Táto oblasť je podľa mňa miestom, kde sa šetriť neoplatí. Za kvalitu sa užívateľom poďakujú ich nervy i celkový duševný stav. Každý výrobca sa snaží dosiahnuť najvyššiu možnú kvalitu svojich „digitálnych“ produktov. Vypočítaví výrobcovia (pravdaže, ak nie sú monopolom), ktorí vyrábajú lacný a nekvalitný softvér sa bez úplatkov a klamlivej reklamy nemôžu presadiť.

Dnes, v dobe, kedy existuje veľmi veľa kvalitného softvérového vybavenia dostupného zadarmo, sú výrobcovia komerčných produktov nútení ponúkať minimálne rovnakú kvalitu, ako riešenia zadarmo. Ďalšou možnosťou je už iba výroba programov, ktorých funkcionalitu neobsahuje žiadne riešenie dostupné zadarmo.

Ale naspäť k definícii kvality. V softvérových výrobkoch sú minimálne dva odlišné pohľady na ich kvalitu. Jeden je používateľský a druhý je programátorský. Používateľský definuje kvalitu za pomoci použiteľnosti, ergonómie a funkcionality. Programátorský pohľad je však bližší iba tým, ktorí programujú. Kvalitu z používateľského pohľadu vie ohodnotiť ako používateľ, tak i programátor, ale kvalitu z programátorského pohľadu vie ohodnotiť iba programátor, poprípade nástroj na hodnotenie kódu. Programátorská definícia kvality je o prehľadnosti kódu, jeho udržiavateľnosti, efektívnosti a čistote štýlu. Oba pohľady na kvalitu sú dôležité. Treba však pri manažovaní projektu nájsť vhodný pomer vynaloženého úsilia a času na zabezpečenie oboch kvalít.

Užívateľskú kvalitu je možné kontrolovať a zvyšovať za pomoci testovania, ktorému sa v tejto eseji budem bližšie venovať. Kvalitu kódu je možné kontrolovať za pomoci statických testov, teda prehliadok kódu, alebo automatizovane za pomoci nástroja Cccc [2] a iných. Kvalitu kódu je možné následne zlepšiť zmenou architektúry softvéru a „preprogramovaním“ relevantných častí (pri OOP paradigme), alebo klasickým refaktoringom.

Testujem, testuješ, testujeme...

Pod pojmom testovanie si človek nezasvätený do problematiky predstaví len jednu časť tohto procesu. Testovanie sa v skutočnosti ale odohráva (malo by sa) takmer vo všetkých fázach vývoju softvéru. Je jedno, či ide o extrémne programovanie,

inkrementálny vývoj, alebo vodopádový model projektu. Testovať sa má napríklad už i samotná špecifikácia projektu. Hľadajú sa staticky (čítaním) chyby, poprípade sa pokúšame v neskorších fázach vývoja namodelovať samotný softvér za pomoci jazyku Z (v praxi sa používa zriedka) a takto otestovať konzistentnosť návrhu. Konzistencia špecifikácie a návrhu je nutnou podmienkou pre bezproblémové implementovanie systému.

Ak statické testovanie predchádza dynamické testovanie, je podľa [3] odhalených viac chýb. Toto sa môže na prvý pohľad pre laika zdať ako nežiadúci efekt. Ale nie je tomu tak. Cieľom testovania je nájsť čo najviac chýb v produkte, lebo sa tým zmenšuje počet neodhalených chýb. Cieľom testovania nikdy nemôže byť preukázanie, že program je bez chýb! Toto je veľmi dôležité a testujúci pracovník si to musí uvedomiť pri vykonávaní testov. Cieľom je overiť funkcionálnu i nefunkcionálnu časť produktu a preukázať, na akej úrovni je. Popri testovaní sa snažíme nájsť čo najviac chýb a pridať produktu na kvalite. Samozrejme, nesmie sa zabudnúť na odstránenie všetkých nájdených chýb a následnom opakovaní všetkých testov.

Ja sa zaoberám v tejto práci testovaním vo fáze, v ktorej ho väčšina z Vás pozná. Ide o testovanie softvérového produktu vo fáze implementačnej. Testovanie v tejto fáze projektu sa delí na statické a dynamické. Statické testovanie je prezeranie kódu vizuálne a hľadanie logických chýb, chýb, ktoré sa vniesli pri kopírovaní častí kódu, a podobne. Pre toto testovanie je vyvinutých i pár nástrojov, ktoré napríklad kontrolujú, či programátor v danom kóde uvoľňuje alokovanú pamäť v každej z možných vetiev programu. Existujú i nástroje hľadajúce opakujúce sa časti kódu, na ktoré následne upozorní vývojára. Podľa práce [3] by malo statické testovanie predchádzať testovaniu dynamickému. Dynamické testovanie je možné prevádzať, až keď je hotová dostatočne veľká časť implementačných prác na skompilovanie použiteľnej časti softvéru. Teda, pri dynamickom testovaní sa skompiluje funkčný produkt, alebo jeho časť a tá sa „testuje“. Najprv sa ide podľa používateľskej príručky a kontroluje sa funkčnosť produktu. Neskôr sa skúšajú rôzne menej štandardné spôsoby použitia, ktoré by mali tiež fungovať. V tejto fáze sa môže testovať i použiteľnosť, familiárnosť a iné aspekty produktu, ktoré sú dôležité pre zákazníka.

Testovanie statické

Statické testovanie sa niekedy nazýva statická analýza kódu alebo inšpekcia kódu (i keď analýza kódu a inšpekcia majú mierne odlišný význam). Statické testovanie je testovaním, kedy sa programový kód nevykonáva fyzicky, ale iba sa prezerá. Prezerať ho môže buď programátor, ktorý daný kód písal, alebo nezainteresovaná osoba. Ak si kód prezerá jeho tvorca, je vhodné, aby tak robil s odstupom času. Toto pravidlo pri poradí testovania najprv statické a následne dynamické, predurčuje nutnosť nezainteresovaných programátorov.

Chyby objavené týmto testovaním je jednoduchšie a najmä lacnejšie opravovať, ako chyby nájdené akýmkoľvek iným spôsobom. Cena opravy chyby je závislá na tom, kedy sa nájde, teda, ako má programátor v hlave ideu daného kódu a koľko nákladov treba vynaložiť na nájdenie, úpravu a znovunasadenie opravených verzií softvéru. Ak

statické testovanie prebehne ihneď po ukončení programovania modulu, tak je táto chyba veľmi rýchlo opraviteľná a položka znovunasadenia programu je nulová.

Ak je čitateľom kódu cudzia osoba, jedná sa z pohľadu čiernej a bielej skrinky o testovanie čiernej skrinky. Je nutné, aby sa budúci čitateľ kódu oboznámil so špecifikáciou, ktorú má kód spĺňať, alebo poznať aspoň požiadavky na základné informácie o vstupe. Nemá však poznať vnútro riešenia. Teda algoritmus kódu si prečíta a následne ho pochopí, čo nakoniec vyústi do toho, že sa pre neho premení čierna skrinka na skrinku bielu. Pri poznaní iba požiadaviek na vstup je možné však iba skontrolovať, či daný kód akceptuje a bez chyby spracuje korektné vstupy. Či je výstup v poriadku a v súlade so špecifikáciou, nemôže však čitateľ zistiť. Na to je potrebné byť oboznámený so špecifikáciou. Tu je ďalšia výhoda, keď je kód čítaný nezainteresovanou osobou. Čitateľ sa zoznámi so špecifikáciou. Je to teda minimálne druhá osoba, ktorá si danú špecifikáciu prečíta a pochopí ju. V prípade, že predošlý čitateľ (programátor) pochopil niečo v špecifikácii inak, ako „kontrolór“, pri prehliadke kódu sa takáto chyba objaví.

Statická kontrola kódu čítaním však nie je zameraná na hĺbkovú analýzu kódu čitateľom. Cieľom je nájsť logické chyby, preklepy, chyby pri práci so vstupom, alebo chyby pri práci s výstupom daného kódu. Kontrola záludnejších chýb je jednoduchšia za pomoci dynamického testovania, kedy sa chyba prejaví a zistí sa tak i približné miesto chyby, kde sa potom táto chyba hľadá pri dôkladnom čítaní kódu, alebo ladení (debug).

Testovanie dynamické

Dynamické testovanie je testovanie aplikácie za behu. Teda skompilovanie zdrojového kódu a následné spustenie daného kompilátu a jeho testovanie sa nazýva dynamické testovanie. Tento typ testov programu sa vykonáva s rôznymi cieľmi. Hlavným cieľom je kontrola funkčnosti podľa špecifikácie a hľadanie chýb. Ďalšími možnými dôvodmi pre dynamické testovanie je testovanie výkonnosti. Teda analýza „úzkeho hrdla“ aplikácie. Testovanie za cieľom merania jeho výkonu je podľa mňa druhý najčastejší dôvod vykonávania dynamického testovania.

Najjednoduchšie dynamické testovanie prebieha vo výrobe. Skompiluje sa funkčný celok aplikácie, tester ho spustí a podľa predlohy kontroluje funkcionalitu daného diela. Tento scenár je však dosť idylický. Takéto testovanie by bolo pre spoločnosť príliš drahé. Zamestnať dostatočný počet kvalitných testerov by bol problém číslo dva. V praxi sa využívajú rôzne programy, ktoré po predložení predpisu (skriptu) ako testovať, otestujú danú aplikáciu a vrátia výsledok. Jednoduché operácie tak tester uloží do skriptov a vždy pri novej verzii vykoná automatické testovanie a až následne po odobrení v automatickom teste sa tester pustí do manuálneho testovania funkcionality, ktorá nemohla byť skontrolovaná automaticky. Pri automatickom testovaní však hrozí, že v samotnom testovacom programe sa nachádza chyba, ktorá spôsobí vyhodnotenie neúspešného testu ako úspešné. Ďalšou hrozbou je, že tester napíše skripty pre testovanie nesprávne, a takisto výstup z testovacieho nástroja nebude

správny. Tieto riziká sa dajú minimalizovať používaním overeného nástroja a zaškolením testera. Tester počas svojej praxe nadobúda dôležité schopnosti vytušiť správny spôsob, ako otestovať čo najväčšiu časť funkcionality a preto je dobré rolu testera nechať na pracovníkoch, ktorí budú na danom poste dlhšiu dobu.

Porovnanie kvality

Je možné porovnanie kvality? Pri výrobkoch hmotných, na ktoré sú na trhu už dlhé generácie sa podarilo vyvinúť rôzne mechanizmy a spôsoby porovnávania kvality. Recenzie, testy ale i porovnania rôznych výrobkov v renomovaných skúšobniach udávajú výhody a nevýhody rôznych výrobkov. Pri softvéry by sa dalo povedať, že je možné spraviť to isté. Z časti to je pravda. Nie je problém porovnať zoznam funkcií jedného programu a tak zistiť, ktorá aplikácia, či softvérový balík má navrch. To je pravda. Ale skúsme si takto modelovo porovnať program VirtualDub s programom MEncoder. Oba majú porovnateľné možnosti v konverzii video súborov. Zdalo by sa až, že je pre zákazníka jedno, ktorý z nich použije. Oba sú zadarmo a funkcionalita rovnaká. Háčik je však v ovládaní. MEncoder je klasická konzolová utilitka typu „švajčiarsky nožík“ a program VirtualDub je extra jednoduchý, ale grafický, lineárny strihový editor.

Pre porovnanie kvality softvéru (vlastne akéhokoľvek tovaru) sa vyvinulo a vyvíja niekoľko štandardov, metrík a spôsobov merania kvality. Vo svete kvality softvéru však nie je žiaden zo štandardov nejako zvlášť rozšírený a známy. Softvér sa na kvalitu z používateľského hľadiska prakticky netestuje, aspoň nie verejne. Po zavedení štandardu ISO 9000 si mnoho ľudí mylne myslelo, že toto bude štandard kvality akýchkoľvek výrobkov. Štandard ISO však nie je o kvalite výrobkov, ale o kvalite výrobného procesu, kvalite kontroly kvality, kvalite manažmentu a riadenia a kvalite dokumentácie. Nájdu sa však i pochybnosti o ISO norme [4]. Ja osobne som zažil audit kvôli ISO norme a môj názor je, že je to len spôsob, ako minúť nadbytočný zisk. Spôsob práce a poriadok na pracovisku, aký bol pri audite vyžadovaný mi pripomínal skôr kanceláriu, v ktorej je zákaz pracovať, a nie tvorivú dielňu, kde by sa mal konať aktívny vývoj ako hardvéru, tak i softvéru.

Budúcnosť je svetlá

Budúcnosť však naznačuje, že sa problematika kvality podstatne viac preskúma a ujednotí. Už len stretnutia Workshop on software quality (WOSQ) naznačujú, že sa v tejto oblasti vykonáva výskum a bádanie, ktoré bude nasledované určite vývojom produktov skvalitňujúcich prácu v tejto oblasti. Výsledky nemožno čakať ihneď, veď informatika je ako veda veľmi mladá, ale časom sa určite i do oblasti kvality softvéru zavedú pevné štandardy a pohľady, ako je ten na obrázku 1. nám budú cudzie.

Obr. 1. Nepotešujúci pohľad na výsledok chyby v softvéri

Použitá literatúra

1. Armour, P. G.: *The Unconscious Art of Software Testing*, COMMUNICATIONS OF THE ACM (January 2005/Vol. 48, No.1).
2. Dustin, Elfriede: Effective software testing : *50 specific ways to improve your testing* (Addison-Wesley, December 2002).
3. Gustav Evertsson: *Inspection vs. Testing*, Blekinge Institute of Technology, Software Verification and Validation (November 2002).
4. Jim Wade: viewpoint : *Is ISO 9000 really a standard?*, ISO Management Systems (May-June 2002). Dostupné na <http://www.bin.co.uk/IMS_May_2002.pdf>

Annotation

Quality of software and its verification

Quality. What this word stands for? In this essay I am trying to describe techniques of measuring software quality by making tests on its. This document is just light introduction to this problem area. It contains description of dynamic testing and static testing. Comparison of some testing techniques is also described here. Assurance of quality is now days very discussed and some standard is needed. Last described problem in this essay is problem between automated tests and manual testing of software. Key to the success is to find equilibrium in ratio of automated (also called scripted) tests and manual testing.