Proces testovania softvéru

BÁLINT FARKAS

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

Abstrakt. Čoraz silnejšia konkurencia na softvérovom trhu, stále stúpajúce požiadavky od používateľov a dynamický vývoj softvéru spôsobuje vytváranie čoraz kvalitnejších a dokonalejších softvérov. Následkom toho, zabezpečenie kvality softvéru dostáva dôležitejšiu rolu , čo umožňuje, aby bol projekt kompletne vykonávaný v súlade s vopred stanovenou špecifikáciou, štandardmi a požadovanou funkcionalitou, bez chýb a možných problémov. Testovanie softvéru je proces slúžiaci na overenie, či softvér zodpovedá požiadavkám ako aj procesom odhaľovania chýb. Cieľom je dosiahnutie požadovanej kvality softvéru z hľadiska funkčnosti, použiteľnosti, spoľahlivosti a výkonnosti. V tejto eseji sa venujem samotnému procesu testovania, uvádzam základné atribúty kvalitného softvéru, opisujem spôsoby a stratégie efektívneho testovania softvérového produktu.

Úvod

V dnešnej dobe, v tomto dravom svete, každý z nás dennodenne používa nejaký softvér. Väčšina ľudí si nevie ani predstaviť, aké úsilie je potrebné na vytvorenie kvalitného softvéru. Dosiahnutie kvality je jedným najdôležitejším cieľom vývoja softvéru. Vytvoriť taký produkt, kde nie sú chyby, podľa mňa nie je možné. Sme ľudia, takže robíme chyby. V niektorých oblastiach, kde sa vyskytnú problémy v softvéri, majú nenapraviteľné dôsledky. Ako príklad môžem uviesť leteckú dopravu. V prípade poruchy alebo problémov s používaním softvéru počas letu, môže dôjsť k tragickým nehodám. Týmto som chcel zdôrazniť, akú dôležitú rolu hrá zabezpečenie kvality v softvérovom inžinierstve. Jedna z metód zabezpečenia kvality softvéru je testovanie.

Kvalitný softvér

Vlastnosti kvalitného softvéru môžeme rozdeliť do dvoch skupín, externé a interné atribúty kvality. Externé atribúty sú tie, ktoré sú viditeľné pre zadávateľa

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-7.

a používateľa. Interné atribúty sú neviditeľné pre vonkajšieho pozorovateľa, avšak podstatné pre dosahovanie externých atribútov.

Externé atribúty:

- spoľahlivosť: správanie sa výrobku pri výpadku – výrobok by nemal pri výpadku systému spôsobiť ani fyzické ani ekonomické škody.
- správnosť: softvér pracuje podľa špecifikácie.
- flexibilnosť: ak sa niečo zmení (požiadavky), softvér sa musí prispôsobiť.
- znovupoužiteľnosť: možnosť použiť softvér (jeho časti) v iných podobných aplikáciách.
- kompatibilnosť: možnosť použiť softvérový produkt s inými produktmi.
- efektívnosť: schopnosť minimalizovať požiadavky na hardware, splnenie kritérií spojených so samotným vývojom výrobku.
- prenosnosť: jednoduchosť prenesenia softvéru na inú softvérovú alebo hardwarovú platformu.
- jednoduchosť použitia: miera námahy potrebnej na zaškolenie a používanie softvéru; vrátane jeho inštalácie a správy.

Všetky tieto atribúty sú len rámcovým vymedzením vlastností. Vždy treba popísať, konkrétne čoho sa týkajú. Napr. program môže byť ľahké prispôsobiť zmeneným požiadavkám na vzhľad (farbu, font) okien, nie však na zmenu v požadovanej štruktúre údajov a naopak, program môže byť ľahko používateľný začiatočníkom, experta však môže zbytočne zdržiavať.

Interné atribúty:

- modulárnosť (vhodná architektúra): podstatným spôsobom vplýva napr. na flexibilitu (umožňuje isté druhy zmien, iné zase sťažuje), ale aj efektívnosť, prenosnosť, správnosť, znovupoužiteľnosť.
- zrozumiteľnosť kódu ovplyvňuje flexibilitu/prenosnosť, správnosť.

Čo je testovanie?

Testovanie je proces odvodenia určitých vlastnosti výrobku na základe výsledkov použitia, prevádzky výrobku (vykonania softvérového systému) v známom prostredí s vybratými vstupmi [3]. Z iného pohľadu, testovanie je proces používaný na zistenie správnosti, úplnosti, bezpečnosti a kvality vyvíjaného softvéru. Z tohto pohľadu nemôže testovanie nikdy zaistiť úplnú správnosť softvéru. Inými slovami, testovanie môžeme chápať ako kritiku alebo porovnanie aktuálnej hodnoty s očakávanou.

Testovania softvéru

Proces „testovanie“ je štandardnou súčasťou životného cyklu softvéru. Nie je možné ho vynechať a keď to robíme naozaj dôkladne, tak patrí medzi najnáročnejšie činnosti. Zvyčajne zaberá až 40 % z celkového vývoja softvéru. Najdôležitejším cieľom testovania je objavenie chýb v programe. Tester sa snaží odhaliť slabé miesta v programe, program „zbúrať“, priviesť do nestabilného stavu, vyvolať výnimky a pod. Výsledkom dobrého testovania je odhalenie čo najväčšieho počtu chýb, ktoré sú následne opravené.

Najprv by bolo rozumné predviesť definíciu testovacieho prípadu [2]. Definícia testovacieho prípadu obsahuje nasledujúce položky.

- vstupné dáta: dáta zadávané na vstupe testovaného programu,
- očakávané výstupné dáta,
- popis zmyslu testu.

Zadaním vstupných dát do programu získame výstupné dáta, ktoré porovnávame s očakávanými výstupnými hodnotami. Pokiaľ sa nezhodujú (vyskytla sa chyba) hovoríme o tom, že testovací proces bol úspešný.

V tvorbe dát pre testovacie prípady existujú dve základné techniky.

- biela skrinka
- čierna skrinka

Jednou z primárnych príčin slabého testovania je fakt, že mnoho programátorov začína nesprávnou definíciou samotného testovania. Vychádzajú z nasledovných úvah:

- Testovanie je proces, ktorý ukáže, že v programe nie sú chyby.
- Cieľom testovania je ukázať, že program vykoná požadované funkcie správne.
- Testovanie je proces, ktorý vytvorí presvedčenie, že program robí to, čo sa od neho očakáva.

Tieto definície sú ale nesprávne. Testovaním chceme softvéru pridať určitú hodnotu. Pridaná hodnota pomocou testovania znamená zvýšenie kvality a spoľahlivosti programu, pričom zvyšovanie spoľahlivosti znamená hľadanie a odstraňovanie chýb. Z tohto dôvodu netestujeme programy preto, aby sme dokázali, že pracujú správne, ale mali by sme vychádzať z predpokladu, že program obsahuje chyby a skúsiť nájsť z nich čo najviac. Vhodnejšia definícia je potom:

Testovanie je proces prevádzania testovacích výpočtov s úmyslom nájsť chyby [4].

Ľudia sú cieľavedomí, s úmyslom dosiahnuť to čo si stanovia a stanovením primeraných cieľov dosiahnu významný psychologický efekt. Keď je naším cieľom dokázať, že program nemá chyby, nemáme takú motiváciu a pravdepodobnosť nájdenia chýb sa zníži. Naopak, keď je naším cieľom ukázať, že softvér obsahuje chyby, je väčšia pravdepodobnosť, že naše testovacie dáta zistia chyby. Druhý spôsob,

ako spevniť definíciu testovania je používanie slova „úspešný“ a „neúspešný“ hlavne v konkrétnych prípadoch, keď projektový manažér zaraďuje výsledky svojich testovacích prípadov. Test, ktorý neodhalí chybu volajú ako „úspešný test“ a ten, ktorý odhalí nedostatky ako „neúspešný test“ . Testovaním je prakticky nemožné ukázať, že chyba sa v programoch nevyskytuje, a to aj vo väčšine triviálnych programov. Väčší pokrok je možné očakávať v počiatočnej fáze testovania a čím je dlhší proces testovania, tým menší pokrok je možné sledovať. Aj podľa tretej uvedenej definície „Testovanie je proces, ktorý vytvorí presvedčenie, že program robí to, čo sa od neho očakáva“, aj keď softvér splní požiadavky používateľa môže obsahovať chyby. Tie sa môžu prejaviť neskôr v kritických situáciách. Ako to Murphy napísal „Ak systém môže ‚spadnúť‘, spadne – a to v najhoršom (najnevhodnejšom) možnom okamihu.“

Ako na testy efektívne?

Odhadnúť rozumnú mieru testovania je takmer nemožné a preto je správne neodhadovať, ale merať a počítať. Základom pre definíciu správneho rozsahu a spôsobu testovania sú nasledujúce veličiny [5]:

- Kritičnosť aplikácie: celkom inú kvalitu testovania vyžaduje inteligentný systém pre zariadenie bŕzd v sériovo vyrábanom aute, kde chyba môže spôsobiť vážne nehody alebo smrť, zase inú úroveň vyžaduje interný firemný informačný systém alebo webová aplikácia.
- Typ aplikácie: bankové systémy sú kritické z celkom iného hľadiska ako telekomunikačné. Zatiaľ, kým pri prvom je najdôležitejšia bezpečnosť a sledovateľnosť, pri druhom je to stabilita a prevádzková spoľahlivosť.
- Zložitosť aplikácie a jej časti: dôkladnosť testovania je potrebné prispôsobiť k zložitosti jednotlivých častí aplikácie a k pravdepodobnosti chýb. To vyžaduje skúsenosť a znalosť chybovosti.
- Znalosť tímu a jeho história: iné sú kontroly výstupu od nového pracovníka, ako od dlhoročného zapracovaného špecialistu, u ktorého je známe, ako pracuje. Znalosť je dôležitá. Chybovosť jednotlivých autorov sa často líši rádovo aj po niekoľkých rokoch práce.
- Predchádzajúce skúsenosti z testovaní: skúsenosti umožňujú naplánovať priemernú dĺžku testovania a testovacie postupy, pretože už v určitých podmienkach boli vyskúšané a overené. Napríklad môžeme preukázať, že pre určité typy úprav regresné testy nie sú efektívne, pretože náklady na odhalenie chýb sú pri nich vyššie, ako pri lokálne pretestovaných, menených moduloch.
- Dostupné prostriedky a nástroje: napríklad pripravenosť dát alebo automatizované nástroje môžu regresné testy natoľko uľahčiť, že ich realizácia bude výhodná aj pri menších zmenách. Naopak, pre tím, ktorý

nepredpokladá vývoj opakovaných aplikácií, bude investovanie do nástrojov a prípravy regresných testov neefektívne.

Stratégie testovania

Testovanie by malo byť vopred naplánované spolu s celým softvérovým procesom. Pre zvýšenie efektivity testovania je vhodné prevádzať aj inšpekciu kódu. Testovanie by malo začínať na úrovni jednotiek (procedúry, triedy – každé overiť samostatne) a postupovať smerom k väčšiemu celku (podsystému, celému systému) [1].

Testovanie jednotiek často robí ten, kto danú časť napísal. Testovanie väčších celkov vykoná alebo aspoň riadi špecialista – tester (pri rozsiahlych projektoch nezávislá testovacia skupina).

Pre väčšinu programátorov je obťažné testovať vlastný program, pretože ich záujem je skôr ukázať, že ich program neobsahuje defekty a pracuje podľa požiadaviek zákazníka. Inými slovami, pretože testovanie je deštruktívny proces , pre programátora môže byť veľmi obťažné prepnúť z kódovaní na testovanie. Program môže obsahovať chyby spôsobené nepochopením špecifikácie, ktoré programátor nemôže sám odhaliť.

Testovanie má prebiehať súčasne s implementáciou systému v nasledujúcich krokoch:

- Testovanie jednotiek (unit testing) – testujeme najmenšie jednotky návrhu, napr. procedúry alebo funkcie, na testovanie môžeme používať bielu-skrinku techniku.
- Integračné testovanie (integration testing) – tým testujeme defekty tykajúce sa rozhraní.
- Validačné testovanie (validation testin) – funkcie viditeľné používateľom.
- Testovanie systému (system testing) – testujeme na úrovni celého systému. Na tvorbu testovacích scenárov sa tu využívajú skutočné scenáre, ktorých vykonávanie sa očakáva od vyvíjaného systému.

Testovanie jednotiek

Pojmom „jednotka“ sa v prípade konvenčne napísaného softvéru zvyčajne myslí procedúra, funkcia, alebo najmenšia samostatne preložiteľná jednotka zdrojového kódu. Jednotka sa testuje samostatne, okolité jednotky sú nahradené ovládačom testu (riadiacou testovanou jednotkou) alebo testovacími maketami. Používa sa technika bielej-skrinky.

Pre objektovo orientované programy sa za jednotku považuje trieda. Triedy ako samostatné komponenty sú zvyčajne rozsiahlejšie ako samostatné podprogramy. Pri testovaní by sme mali samostatne otestovať každú metódu. Niektoré metódy sa dajú testovať až po vyvolaní predchádzajúcich metód, napr. inicializácia objektu. Pokiaľ používame dedičnosť, musíme testovať aj všetky zdedené operácie (môžu obsahovať predpoklady o ďalších operáciách a atribútoch ktoré môžu byt potomkom zmenené.

Obdobne musíme znovu otestovať potomka pri zmene rodiča). Je potrebné testovať nastavenie všetkých atribútov objektu, testovať prechod všetkých stavov objektu, prípadne simulovať všetky udalosti, ktoré spôsobujú zmenu stavu objektu.

Integračné testovanie

Po otestovaní individuálnych komponentov musíme komponenty integrovať – zostaviť do čiastočného alebo úplného systému. Výsledok musíme otestovať na problémy, ktoré vznikajú pri interakcii komponentov. Jediný možný prístup je, že po otestovaní čiastkových modulov zostavíme aplikáciu, ktorá je ale použiteľná iba pre malé programy. Pre väčšie systémy je to najmenej efektívny spôsob integrácie.

Hlavným problémom je lokalizácia defektov, pretože vzťahy medzi komponentmi môžu byt značne zložité. Preto sa často pre integrácie a testovanie používa inkrementálny prístup. Najprv integrujeme minimálnu konfiguráciu systému a otestujeme. K systému pridávame inkrementy a po každom pridaní systém otestujeme. Keď nastane problém, bude pravdepodobne spôsobené pridaním posledného inkrementu.

V skutočnosti to nebude také jednoduché, pretože niektoré vlastnosti budú rozptýlené po niekoľkých komponentoch, vlastnosť môžeme otestovať až po integrácii týchto komponentov. Pri plánovaní testu treba počítať s časovým plánom na dokončenie modulu. Pokiaľ má dôležitý modul neočakávané problémy, môže sa tým zdržať cela integrácia (programátor rieši problém, kým ostatní naňho čakajú).

Testovanie rozhraní

Cieľom testovania rozhrania je detekovať defekty, ktoré môžu vzniknúť chybnou interakciou medzi modulmi alebo podsystémami alebo chybným predpokladom o rozhraní. Testovanie rozhraní je obtiažne, pretože niektoré defekty sa prejavujú iba v nezvyčajných podmienkach. Odporúčania pre testovanie rozhraní:

- V testovanom kóde nájdite všetky volané externé komponenty.
- Navrhnite množinu testov tak, aby externé komponenty boli vyvolané parametrami, ktoré sú extrémy ich rozsahu (napr. prázdny reťazec, dlhý reťazec, ktorý môže spôsobiť pretečenie a pod.)
- Navrhnite testy, ktoré môžu spôsobiť neúspech externých komponentov
- Navrhnite testy, v ktorých sa bude líšiť poradie aktivácie komponentov.

Validačné testovanie

Začína tam, kde končí integračné testovanie. Testujeme, či softvér spĺňa požiadavky používateľa. Akceptačným testovaním zadávateľ určí, či produkt spĺňa zadanie. Testuje sa na reálnych dátach. Pre generické produkty nie je väčšinou možné vykonať akceptačné testovanie u každého zákazníka, preto prebieha alfa a beta testovanie.

- Alfa testy: Na pracovisku, kde sa softvér vyvíja (známe prostredie). Testuje používateľ, vývojový pracovníci ho sledujú a zaznamenávajú problémy.

- Beta testy: testujú vybraní používatelia vo svojom prostredí (vývojárom neznáme). Defekty ohlásené používateľmi sú opravené a vzniká finálny produkt.

Záver

Testovanie je štandardnou súčasťou životného cyklu softvéru a niekedy patrí medzi najnáročnejšie činnosti. Spôsob, úspešnosť a efektivita testovania dosť v značnej miere ovplyvňuje celkovú kvalitu softvéru a preto ho nemožno podceňovať. V tomto článku som kategorizoval základné vlastnosti, čo sa týka kvality softvéru. Uvádzal som hlavné stránky efektívneho testovania a podrobnejšie som rozpísal jednotlivé kroky (stratégie) testovania, čo by sme mohli využívať aj vo Vašom tímovom projekte.

Použitá literatúra

1. BERTOLINO, A.: Software Testing Research: Achievements, Challenges, Dreams. In: International Conference on Software Engineering, 2007 Future of Software Engineering, IEEE Computer Society, Washington, DC (2007), 85-103.
2. BEIZER, B.: Software Testing Techniques, Second Edition, ISBN: 1850328803, The Coriolis Group
3. BIELIKOVÁ, M.: Softvérové inžinierstvo : Princípy a manažment. 1. vyd. Bratislava: STU v Bratislave, 2000. 220 s. ISBN 80-227-1322-8
4. HARAZÍM K. - MICHNA R. : Trendy v testování a verifikace programů, http://axpsu.fpf.slu.cz/~sos10um/trendy/1-verifikace.doc, Navštívené dňa 14.10.2008.
5. YUEH CHEN, T. - KUO, F – ZHOU, Z.: An effective testing method for end-user programmers. ACM SIGSOFT Software Engineering Notes 30(4): 1-5 (2005)

Annotation

Software testing process

Increasingly competition on software market, rampant requirements from users and dynamic software development, cause creating more qualitier and perfect software. In consequence of thereof, software quality assurance has an important role, what allows, that project was completely exercised conformable with predetermined specification, standards and required functionality, with no errors and possible problems. Software testing is process serve to verification, whether the software is responsible to request as well as process to detecting errors. The goal is be achieved the required software quality in term of functionality, usability, reliability and performance. In this essay I present the process testing, I am leading main attributes of quality software; I describe ways and strategy of effective software product testing.