Vedomie a podvedomie pri testovaní

ĽUDOVÍT LUČENIČ

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

Abstrakt. Esej pojednáva o psychologických aspektoch v softvérovom procese pričom sa zameriava najmä na fázu verifikácie a validácie. Na odhaľovanie chýb a zlyhaní možno pozerať napríklad z pohľadu dynamiky, automatizácie, testujúceho subjektu či návykov používateľov i testerov. Jednou z kľúčových úloh testovania je aj zistenie, či boli splnené nefunkcionálne požiadavky. Testovanie a prehliadky (dynamika) napriek tomu, že predstavujú odlišné techniky, nemožno od seba odlúčiť. Esej sa ďalej venuje porovnaniu automatizovaného resp. manuálneho prístupu k testovaniu i porovnaniu pohľadov tvorcu kódu a používateľa. Sú spomenuté ich rôzne návyky, ktoré majú vplyv na výslednú kvalitu resp. výslednú spokojnosť používateľa. Pre účely manažovania projektov je potrebné zohľadniť tieto psychologické aspekty pri plánovaní (testovania) i pri z toho vyplývajúcich odhadoch nákladov. Nemenej dôležitým je aj riziko vzniku konfliktov medzi testermi a programátormi, ktoré treba efektívne riešiť alebo sa mu pokúsiť predchádzať.

Úvod

Akákoľvek ľudská činnosť, či už je umelecká, systematicky tvorivá alebo monotónne mechanická je ovplyvnená niektorou zložkou ľudskej osobnosti – nadvedomím, vedomím alebo podvedomím. Je dôležité si túto skutočnosť plne uvedomiť pri snahe o zefektívnenie ľudskej práce, spolupráce a procesov, ktoré s ňou súvisia. Softvérový proces je proces riadený a vykonávaný ľuďmi, a preto je nutné rozmýšľať o ňom i v načrtnutom kontexte psychológie. Dôležitou fázou v softvérovom procese, ktorý má za cieľ vytvoriť kvalitný produkt (t.j. uspokojiť zákazníka), je bezpochyby fáza verifikácie a validácie, zjednodušene testovanie. Na verifikáciu a validáciu možno nazerať z pohľadu dynamiky, automatizácie či napríklad podľa subjektu, ktorý ju vykonáva (tester/tvorca kódu, resp. používateľ) pri súčasnom zohľadnení jeho konkrétnych návykov. Pri testovaní napríklad (ako vyplýva z praxe – [1]) s postupným získavaním skúseností konkrétneho testera dochádza k pozvoľnému prechodu od vedomej činnosti k podvedomej. Takisto na úrovni ľudských vzťahov (konflikty medzi testermi, programátormi, príp. používateľmi) rovnako ako i na úrovni manažmentu

Manažment v softvérovom inžinierstve, máj 2005, s. 1-8.

zdrojov (odhady nákladov či stanovenie plánov testovania) možno nájsť niekoľko podnetov na zamyslenie. O tom ako zohľadniť tieto jednotlivé aspekty v praxi manažovania softvérových projektov pojednáva táto esej.

Testovanie softvéru z pohľadu dynamiky

Z pohľadu dynamiky existujú dve techniky verifikácie a validácie. Ide o tzv. prehliadky a testovanie. Prehliadky predstavujú statický prístup a zjednodušene si pod nimi možno predstaviť prechádzanie zdrojových textov tvorcom kódu, resp. testerom a vyhodnocovanie jeho kvality (efektívnosť, čitateľnosť, ap.) vzhľadom na (funkcionálne) požiadavky kladené na výsledný softvérový produkt [3]. Testovanie je na druhej strane technika doplňujúca (z pohľadu dynamiky) techniku prehliadok. Jeho pridanou hodnotou je najmä skutočnosť, že dokáže odhaliť stav realizácie nefunkcionálnych požiadaviek. Primárne však prehliadky slúžia na odhaľovanie chýb a testovanie sa používa na odhaľovanie zlyhaní. Zlyhania môžu predstavovať nekorektné dynamické správanie sa systému v dôsledku jednej alebo viacerých chýb v zdrojovom kóde aplikácie.

Ako sa uvádza v [2], bolo empiricky overené, že množstvo chýb odhalených vo fáze verifikácie a validácie pri súčasnom použití oboch uvedených techník závisí od ich nadväznosti. Uvádza sa ďalej, že populárnejší postup testovania s následným uskutočnením prehliadok je paradoxne menej efektívny čo do počtu nájdených chýb v zdrojových textoch. Autor uzatvára, že testovanie by malo nasledovať po prehliadkach a že takýto postup je efektívnejší než spomínaný opačný.

Skúsme sa teda zamyslieť nad psychologickým pozadím uvedenej empiricky objavenej skutočnosti. Snaha každého človeka dosiahnuť čo možno najskôr „hmatateľný“ výstup je pomerne pochopiteľná. Pri postupe, kedy začíname testovaním, sa dá demonštrovať pomerne rýchlo, či systém vykazuje nejakú chybovosť alebo nie. Samozrejme za predpokladu dobre zvolenej stratégie testovania. Možno hádam priamo bez nadbytočných analýz tvrdiť, že z výsledkov prehliadok (i keď nimi možno odhaliť principiálne viac chýb) nedokážeme povedať nič o dynamike systému. Z psychologického hľadiska venovať čas na činnosť (t.j. na prehliadky), ktorá z vonkajšieho pohľadu neposúva systém do praxe používateľa, skôr demotivuje. Testovanie na druhej strane zabezpečuje práve tento posun.

Medzi ďalšie výhody postupu testovania s následnými prehliadkami možno snáď započítať i skutočnosť, že vtedy nastáva istý časový odstup medzi tvorbou samotného kódu a jeho opätovným prechádzaním. Tento argument sa môže javiť kontroverzný, avšak treba si uvedomiť, že časový odstup môže mať za následok uvoľnenie fixných myšlienok spojených s kódom a teda vytvorenie miesta pre myšlienky nové. Argument má váhu najmä v prípade, že tvorca kódu a inšpektor (vykonávateľ prehliadok) je tá istá osoba. Na druhej strane za nevýhodu testovania ako prvej fázy možno označiť napr. potrebu implementovania logicky rozsiahlejšej časti, aby sa nemuselo testovať z globálneho pohľadu viac-menej samoúčelne, ale aby sa dala priamo pri testovaní

overovať napríklad aj miera integrácie rozhraní prípadne ďalšie významné aspekty kvalitného produktu.

Opačný postup (prehliadky nasledované testovaním) môže odhaliť širšie spektrum chýb, k čomu sa dostanem neskôr podrobnejšie. Na tomto mieste spomeňme napríklad chyby v návrhu (neúplnosť), prípadne nedokonale analyzované rozhrania spolupracujúcich objektov ap. Za jeho nevýhody možno v protiklade s prvým postupom považovať málo „hmatateľný“ výstup spočiatku fázy verifikácie a validácie, neskoršie nasadenie (i keď len predprodukčné) a najmä riziko postupného znižovania koncentrácie ľudí pri dlhotrvajúcich prehliadkach.

Spomenuté argumenty sú prehľadne zhrnuté v Tab.1. Sú tam zvýraznené argumenty, ktoré podľa môjho názoru najväčšmi prispievajú k analyzovanej skutočnosti. Pri prehliadkach sa dá relatívne v skorej fáze zvýšiť kvalita zdrojových textov, čo môže mať multiplikatívny efekt vo vzťahu k následnému testovaniu.

Postup	*Testovanie a prehliadky*	*Prehliadky a testovanie*
VÝHODY	• hmatateľný výstup (skoré nasadenie) • časový odstup medzi tvorbou kódu a prehliadkami (?) • väčší dôraz na "dynamiku" systému – t.j. správanie sa v reálnom prostredí (nefunkcionálne požiadavky)	• odhalenie chýb rozhraní • odhalenie chýb návrhu • **skvalitnenie zdrojových textov (napr. čitateľnosť)** • **z toho vyplývajúce zrýchlenie odstránenia implementačných chýb** • efektívnejší postup (viac odhalených chýb) • väčší dôraz na "statiku" systému – t.j. architektúru a návrh
NEVÝHODY	• menej efektívne (menej odhalených chýb) • vyžaduje si ucelenejšie celky kódu, aby testovanie "malo zmysel" • **výsledky testovania upriamia pozornosť len na určitú časť kódu (môžu uniknúť dôležité chyby, na ktoré sa nenarazilo v dôsledku nepokrytia testovacími scenármi)** • **z toho vyplývajúca závislosť od testovacích scenárov**	• malo hmatateľný výstup na počiatku • neskoršie nasadenie • venovanie väčšej pozornosti prehliadaniu kódu môže znížiť sústredenie - efektívnosť klesá • nefunkcionálne požiadavky v úzadí

Tab. 1 Výhody a nevýhody rôznych poradí aplikácie prehliadok a testovania

Zároveň sa vyhneme riziku, že pri testovaní narazíme na určitú časť kódu, ktorá pohltí všetky naše prostriedky a zdroje, a tak sa nám nepodarí zabezpečiť dostatočne diverzifikované testovanie rôznorodými testovacími scenármi pokrývajúcimi maximum kódu.

Testovanie softvéru z pohľadu miery automatizácie

Pri vychádzaní zo základného princípu všeobecného manažérstva kvality, ktorý definuje kvalitný výrobok ako výsledný produkt kvalitného procesu výroby, resp. tvorby, sa dá dospieť od požiadavky na kvalitný výsledok (v našom prípade softvér) až k predpokladu kvalitného procesu testovania. Ten je totiž nevyhnutný k tomu, aby bolo možné prehlásiť softvér za kvalitný. Určiť kvalitu procesu testovania nie je však vo väčšine prípadov jednoduché. Vplýva na ňu viacero parametrov ako napríklad miera automatizácie procesu, jeho plánovanie a v neposlednom rade i vyhodnocovanie a následné priebežné zapracovávanie získaných poznatkov do samotného procesu.

Skúsme rozobrať vhodnú mieru automatizácie procesu testovania softvéru. Najskôr si musíme uvedomiť, že automatizácia sa môže týkať plánovania testovania, jeho vyhodnocovania alebo napríklad definovania scenárov testovania a v neposlednom rade i samotného (*run-time*) testovania. Čo sa má teda rozumieť pod automatizáciou tohto procesu?

Pod automatizáciou testovania v užšom zmysle slova možno chápať používanie testovacej utility, ktorá zabezpečuje v pravidelných intervaloch (napr. po zostavení novej verzie) spustenie sady vopred definovaných testovacích scenárov. Ich výsledok sa môže ďalej spracovávať ručne i automaticky. Tieto scenáre môžu byť taktiež výsledkom automatizovaného procesu atď. Vo všeobecnosti sa dá predpokladať, že automatizovať pri testovaní možno takmer celý proces. V tejto súvislosti je nevyhnutné poznamenať, že v súčasnosti plná automatizácia nedokáže zohľadniť niektoré aspekty ľudskej činnosti akými sú podvedomé konanie či intuícia [1]. Psychologický aspekt sa momentálne darí nahrádzať len technikami umelej inteligencie, ktoré sú však značne výpočtovo náročné a v konečnom dôsledku vždy len deterministické. Z praxe vyplýva, že skúseného tvorcu testovacích scenárov nemožno ľahko nahradiť práve pre jeho nadobudnuté intuitívne zručnosti. Kde je teda miera vhodnej úrovne automatizácie? Záleží to od účelu. Ak máme projekt, ktorý sa pravidelne opätovne uvoľňuje (*release*), prírastky sú malé, iterácie časté, je vhodné testovanie viac automatizovať ako v opačnom prípade. V takomto prípade čas i prostriedky všeobecne vložené do vytvorenia utility prinesú s najväčšou pravdepodobnosťou napokon značnú úsporu celkových nákladov. Proti tejto jednoznačnej výhode stojí fakt, že testovacia utilita sama o sebe predstavuje taktiež softvér (resp. softvérový produkt). Je preto rovnako náchylná na vnútorné (logické) i vonkajšie (prezentačné) chyby. Dá sa s istotou vo všetkých prípadoch nájdených chýb tvrdiť, že vyskytnuvšia sa chyba je chybou testovanej aplikácie a nie testovacej? Nedá. A to je práve to mínus, ktoré treba mať vždy na pamäti pri rozsiahlom testovaní pri využití automatizácie.

Do istej miery kompenzovať túto skutočnosť dokáže azda len znovupoužitie komponent testovacej utility pri návrhu a implementácii testovacej utility pre novší softvér. Za predpokladu, samozrejme, že častým používaním týchto komponent dochádza k ich skvalitňovaniu.

Testovanie softvéru z pohľadu subjektu testovania

Pri testovaní (v širšom zmysle slova) je nutné uvedomiť si najmä kto testuje a tiež pre koho sa testuje. Dva základné pohľady sú pohľad tvorcu kódu a pohľad používateľa aplikácie. Obidva náhľady zároveň možno použiť aj v roli testujúceho i toho, pre koho sa testuje. Pohľad tvorcu kódu je pohľadom interným. Zahŕňa viacero aspektov, ktoré sú pre koncového používateľa neviditeľné alebo nepodstatné. Testovať teda možno napríklad i požiadavky, analýzu problémovej oblasti, návrh jednotlivých softvérových súčiastok, architektúru celého systému, efektívnosť použitých algoritmov, či dostatočnú funkcionalitu rozhraní jednotlivých komponentov. Na druhej strane pohľad používateľský zahŕňa externé aspekty, ako je konzistencia, funkčnosť a účelnosť používateľského rozhrania, možnosti prepojenia s ďalším softvérom, (ne)závislosť na komponentoch tretích strán, prispôsobiteľnosť zvyklostiam používateľa atď. Ak sa testuje a konzumentom výsledkov tohto testovania je tvorca kódu, môže byť dôležitejšia napríklad čitateľnosť kódu (ako dôsledok aplikovania výsledkov testovania) než doba odozvy systému. Pri pohľade používateľa je pochopiteľne naopak podstatnejšia doba odpovede aplikácie. Z uvedeného viac-menej priamo vyplýva potreba oboch prístupov k testovaniu. Nie je možné niektorý z pohľadov vynechať alebo potlačiť na úkor druhého, pretože sa tieto veľmi potrebne dopĺňajú.

Návyky testera (t.j. človeka vykonávajúceho testovanie z pohľadu tvorcu kódu) i používateľa sú predmetom odborných diskusií, v ktorých sa rozoberá i vplyv podvedomia. Ako som uviedol už skôr, v praxi sa dá pri postupnom získavaní skúseností pozorovať presun od vedomej činnosti k podvedomej. Tento súvisí s viac či menej výrazným uplatňovaním intuície pri výbere testovacích scenárov ap. Dôležitou skutočnosťou je pritom neustála potreba cviku, pretože bez cviku niet návyku. Návyk dokonca nie je možné bez cviku ani udržať, z podvedomia sa jednoducho vytratí. Manažéri si musia toto uvedomiť, ak chcú mať vo svojich tímoch kvalitných ľudí na postoch testerov. Je to dôležité z dlhodobého hľadiska, pretože podvedomie nie je možné ovplyvňovať (aspoň zatiaľ) v krátkom čase. Keby sme však chceli na druhej strane túto polemiku zjednodušiť, mohli by sme prehlásiť, že intuícia a podvedomie priamo súvisia s množstvom skúseností. Ako so všetkým i tu bude pravda istotne voľakde uprostred, a preto sú návyky testujúcich osôb prinajmenšom hodné zamyslenia.

Návyky používateľa patria do úplne inej kategórie. Tie človek nenadobúda vedomou činnosť, resp. cieľavedomou aktivitou, ale ako dôsledok používania veľkého množstva rozličného softvéru. Istú podmnožinu týchto návykov tvoria aj tzv. prirodzené preferencie, za ktoré možno označiť napríklad potrebu prehľadnosti rozloženia ovládacích prvkov, primerané množstvo informácie dostupnej v danom okamihu atď. Ostatné návyky patria potom skôr do kategórie „spoločných čŕt“. Preto

kvalitné testovanie z pohľadu používateľa by malo uvedené návyky rešpektovať. Pri tvorbe softvéru ich treba zakomponovať do požiadaviek i do návrhu používateľského rozhrania a napokon pri samotnom testovaní dôsledne sledovať ich implementáciu. Miera, do akej sa toto podarí v softvérovom projekte zrealizovať, určuje prijatie softvéru používateľom, jeho spokojnosť a napokon i celkovú úspešnosť projektu. Je dôležité si však tiež uvedomiť, že každý softvér je jedinečný, a preto je prakticky nemožné uskutočniť úplnú generalizáciu týchto návykov. Softvér totiž po svojom nasadení ovplyvňuje prostredie, do ktorého prichádza a tým spôsobuje vznik nových podmienok, na ktoré nemusí byť schopný v každom prípade dostatočne pružne reagovať, pretože ich vo všeobecnosti nemožno predpovedať. Návyky používateľa neradno teda odhadovať v extrémne veľkom rozsahu bez „živej“ konfrontácie interagujúceho používateľa.

Konflikty pri testovaní

Každý človek, ak teda premýšľa, je od prírody presvedčený o svojej vlastnej pravde. Ak sa však stretnú dvaja ľudia, ktorí nie sú schopní myslieť dostatočne otvorene, aby boli prístupní i pravde toho druhého, dochádza ku konfliktu. Konfliktom sa spravidla nevyhnú ani programátori vo vzťahoch k testerom. Čo je ich príčinou? Najčastejšie programátor je ten, ktorý musí riešiť najzákladnejšie problémy, na ktoré nemyslel nikto pred ním. On je zároveň i ten, ktorý vytvára niečo konečné – niečo „hmatateľné“. Tester je potom človek, ktorý sa obracia na programátora s konkrétnymi chybami, nedostatkami takpovediac „jeho“ výtvoru. Ak teda obe strany nedokážu pristupovať k svojej práci s dostatočným odstupom (odosobnene), nevyhnutne si niektorá z nich bude brániť svoju pozíciu. A najlepšou obranou je útok. Ako sa tomu dá predísť alebo ako možno túto spoluprácu zefektívniť? Ak sa vedie anonymný zoznam nájdených nedostatkov, závad a chýb, ktoré fixuje niekto, kto nemá žiaden kontakt s tým, kto ich odhalil, je pravdepodobné, že dôjde k istému odosobneniu sa pri testovaní či naprávaní chýb. Takáto podmienka je však v priamom rozpore s požiadavkou manažmentu na transparentné priamočiare hodnotenie zainteresovaných vývojárov i testerov. Optimálnu cestu treba hľadať opäť niekde uprostred. Riešenie sa ponúka v efektívnom manažovaní dostupnosti informácií jednotlivým členom tímu, pričom rovnako dôležité sú i dobrý pocit a zainteresovanosť na kvalite výsledku – slovom motivácia. Správne motivovaný človek sa častokrát bez problémov dokáže preniesť ponad malichernosti vyvolávajúce napätie či priamo konflikt.

Manažment testovania

Keď si uvedomíme, koľko rôznych aspektov sa dá identifikovať len v jednej fáze softvérového procesu, ktoré treba zohľadňovať pri jeho úspešnom riadení, ako je napríklad fáza verifikácie a validácie, nadobudneme pomerne slušnú predstavu o celkovej komplexnosti manažmentu v softvérovom inžinierstve.

Z pohľadu dynamiky je pri verifikovaní a validovaní zaujímavá následnosť prehliadok a testovania. Prehliadky predstavujú statickú fázu, testovanie dynamickú. Obe techniky sa preto vhodne dopĺňajú. Poradie ich aplikácie na softvérový systém však ovplyvňuje počet chýb, ktoré sme schopní identifikovať. Príčiny tejto skutočnosti možno nájsť v prvotnom zvýšení kvality zdrojových textov prehliadkami a súčasným nezameriavaním sa na iba určitú časť kódu pri testovaní.

Automatizácia procesu testovania môže v prípade opakovateľne uvoľňovaného softvéru priniesť výrazné zníženie celkových nákladov i zvýšenie produktivity (efektivity procesu). Treba sa však vyvarovať bezmyšlienkovitej aplikácii uvedeného. Bez dôslednej analýzy pôvodu chyby nemožno totiž ani plne automatizovaným procesom nahradiť intuitívne rozhodnutie človeka.

Ďalšou spomenutou oblasťou bolo zohľadnenie testujúceho subjektu. Rovnako ako v prípade súčasnej potreby dynamického i statického prístupu, vyvstáva i tu potreba dvoch navzájom sa doplňujúcich pohľadov. Žiaden z nich (pohľad tvorcu kódu/pohľad používateľa) nie je nadradenejší a nemal by byť preto uprednostňovaný či naopak opomínaný. Za východiskovú pozíciu možno považovať zamyslenie sa nad návykmi testujúcej osoby. Tieto majú v uvedených prípadoch rozdielnu povahu, avšak na spokojnosť s výsledkom na strane tvorcu i na strane zákazníka vplývajú veľmi rovnocenne.

Poslednou načrtnutou problematikou som sa snažil zaostriť pozornosť na existenciu konfliktov. Generálne riešenie konfliktov spočíva v samotných ľudských vzťahoch, ktoré ak sú na vysokej úrovni, riziko vzniku konfliktov veľmi výrazne klesá. Jedným zo spôsobov posilnenia medziľudských vzťahov by mohla byť napríklad dostatočná motivácia zainteresovaných.

Testovanie predstavuje veľmi dôležitú fázu vývoja softvéru. Zaslúži si preto, aby sme mu na úrovni manažmentu venovali patričnú pozornosť a snažili sa zohľadniť čo najširšie spektrum pohľadov, ktoré majú naň zjavný vplyv. Výzvou nám ostáva odhalenie práve tých najdôležitejších v danom kontexte.

Použitá literatúra

1. Armour, P. G.: The Unconscious Art of Software Testing, COMMUNICATIONS OF THE ACM January 2005/Vol. 48, No.1
2. Evertsson G.: Inspection vs. Testing. Blekinge Institute of Technology
3. Inspection definition, The Free On-line Dictionary of Computing: http://dict.die.net/inspection/, (Marec 2005)

Annotation

Conscious and subconscious in testing

Essay deals with psychological aspects in software process, mainly aiming at verification and validation. Defect and failure detection can be viewed from the point of dynamics, automation, testing subject or testers' and users' habits. One key task of testing is also to inquire the

fulfillment of non-functional requirements. Testing and inspections (dynamics) cannot be separated in spite of being different techniques. Essay further concerns with comparison of automated and manual testing approach as well as various habits of the code creator and user. For the sake of project management it is needed to consider these psychological aspects in planning (of testing) and in cost estimations. Not to a lesser extent important is also the risk of conflict among testers and programmers, which is to be effectively dealt with or avoided.