PROCES TESTOVANIA SOFTVÉROVÝCH PRODUKTOV

THE SOFTWARE PRODUCTS TESTING PROCESS

Pavol TANUŠKA, Peter SCHREIBER

Autori: ***Doc. Ing. Pavol Tanuška, PhD., Doc. Ing. Peter Schreiber, CSc.***
Pracovisko: ***Katedra aplikovanej informatiky a automatizácie, Materiálovotechnologická fakulta STU***
Adresa: ***Paulínska 16, 917 24 Trnava***
Email: ***tanuska@mtf.stuba.sk, schreibe@mtf.stuba.sk***

Abstract

V tomto článku sa venujeme samotnému procesu testovania softvérových produktov (bez nástrojov testovania), uvádzame spôsoby testovania, základné aspekty testovania, ako aj softvérové metriky. V závere naznačujeme, aké dôsledky má nedostatočné testovanie.

This contribution deals with the process of software products testing (without testing tools), with the manner of testing, the base aspects of testing and software metrics will be described also. At the end of article are suggested the consequences of inadequate testing.

Key words

testovanie, verifikácia, validácia, spoľahlivosť, metriky, cyklus životný

testing, verification, validation, reliability, metrics, life cycle

Úvod

V súčasnosti môžeme konštatovať, že proces testovania je opisovaný v mnohých publikáciách, avšak hlavne vychádza z praktických skúseností veľkých softvérových firiem. Pri samotnom procese testovania sa softvéroví inžinieri sústreďujú na celý životný cyklus testovania, ktorý začína testovaním funkcií a končí preberacím testovaním a zahŕňa [1, 4, 5]:

- **Testy funkcií a modulov**, ktoré sú vykonávané prevažne softvérovými inžiniermi priamo v etape implementácie.
- **Integračné testy**, čo je vlastne testovanie viacerých modulov súčasne.
- **Regresné testovanie** – testuje sa, či nenastal vedľajší efekt pridaním nového modulu alebo funkcie (zavlečenie chyby).
- **Nezávislé testy** - vykonávané nezávislými externými subjektami.

- **Alfa a beta testovanie** - čo je vlastne testovanie systému v reálnom prostredí. Pri alfa testovaní sa systém testuje bez živých dát. Testuje ho zákazník u vývojára. Beta testovanie používa reálne dáta so sledovaním výsledkov s možnosťou okamžitej nápravy.
- **Systémové testovanie** – je to séria rôznych testov, ktorá preveruje celý systém (HW, SW prostredie, databáza, ľudia, ...). Môžeme sem zaradiť aj testovanie obnovy (Recovery), bezpečnostné testovanie, výkonnostné a záťažové testovanie (Stress). Obdobne sem môžeme zaradiť aj tzv. testovanie citlivosti (Sensititivity testing), ktoré sa snaží objaviť kombinácie dát (v rámci platných obmedzení), ktoré môžu spôsobiť nestabilitu systému.
- **Inštalačné testy** - zahŕňajúce všeobecnú výkonnosť systému, ktorý je prvýkrát nainštalovaný na konkrétnom HW a operačnom systéme.
- **Validačné testovanie** – slúži k overeniu, že softvér spĺňa „rozumné očakávania“ zákazníka, ktoré sú definované v špecifikovaných požiadavkách. Validačné testovanie sa vykonáva metódami black-box.
- **Preberacie testovanie** - je to vlastne posledný míľnik pri testovaní projektu. V prípade úspešného zvládnutia nastáva oficiálne prevzatie projektu zákazníkom.

Chyby môžu byť do aplikácie vnesené v každom štádiu životného cyklu vývoja aplikácie, vrátane testovania.

Podrobnejšie to dokumentuje nasledovná tabuľka [3].

PERCENTUÁLNE VYJADRENIE POČTU VZNIKNUTÝCH CHÝB V ZÁVISLOSTI, KDE BOLI NÁJDENÉ Tabuľka 1

	Kde boli chyby nájdené					
Kde chyby vznikli	Zhromažďovanie požiadaviek a analýza a návrh	Kódovanie a testovanie modulov	Integračné a systémové testovanie	Testovanie zákazníkom a Beta testy	Vyexpedovaný produkt	Spolu
Zhromažďovanie požiadaviek, analýza a návrh	**3,5**	**10,5**	**35**	**6**	**15**	**70**
Kódovanie a testovanie modulov		**6**	**9**	**2**	**3**	**20**
Integračné a systémové testovanie			**6,5**	**1**	**2,5**	**10**
Spolu	**3,5**	**16,5**	**50,5**	**9**	**20,5**	100%

Ak chceme, aby bol produkt úspešný z hľadiska používateľa, musíme akceptovať päť základných aspektov testovania [2, 4]:

Utility (funkčnosť alebo úžitkovosť).
Zvyčajne sa testuje zložitosť ovládania produktu, či produkt uskutočňuje najdôležitejšie funkcie, či je produkt finančne efektívny atď.
Bez ohľadu na to, či je produkt korektný alebo nie, toto sú najdôležitejšie problémy, ktoré by mali byť otestované.

Reliability (spoľahlivosť).
Je veľmi dôležité vedieť, ako často produkt zlyháva a aké dopady môže mať zlyhanie.
Keď produkt zlyhá, je dôležité odhadnúť, za ako dlho sa podarí chybu odstrániť a ešte dôležitejšie je, za aký čas sa podarí odstrániť následky danej chyby.

Priemerný chod softvérových produktov (napr. informačných systémov) medzi dvomi zlyhaniami je asi 6 mesiacov. Ak nastane zlyhanie a proces obnovy (recovery) je dlhší ako 2 dni, môžeme konštatovať, že spoľahlivosť systému je nízka.

Robustness (odolnosť).
Robustnosť je v podstate množina faktorov ako rozsah ovládacích podmienok, možnosť neakceptovateľných výsledkov zo správnych vstupných údajov a ďalšie.
V prípade testovanie na funkčnosť musia výstupné dáta zodpovedať vstupným podmienkam. Pri testovaní robustnosti výstupné dáta, ktoré nezodpovedajú vstupným podmienkam, sú zámerne dané na vstup a testuje sa, ako na ne reaguje produkt.

Performance (výkonnosť).
Produkty pracujúce v reálnom čase sú charakterizované časovými obmedzeniami, ako napríklad odozva na konkrétnu akciu, doba vzorkovania atď. Pri testovaní výkonnosti je veľmi dôležité otestovať práve tie parametre systému, ktoré sú životne dôležité pre jeho činnosť. Napríklad zber dát z reaktora sa vykoná každú desatinu sekundy. Ak by systém nedokázal spracovať a vyhodnotiť tieto dáta v požadovanom časovom intervale, choval by sa ako systém, ktorý nemá dostatočnú výkonnosť.

Correctness (správnosť, korektnosť).
Produkt je korektný, ak vyhovuje stanoveným technickým podmienkam a je nezávislý na použitých zdrojoch.

Spôsob testovania

V súčasnosti je možné konštatovať, že existuje množstvo postupov verifikácie a validácie informačných systémov, avšak nie exaktných. Existuje síce norma ISO/IEC 9126 stanovujúca kritériá kvality softvéru, ale aj v nej sa zdôrazňuje, že doteraz neboli vypracované všeobecne použiteľné postupy ich merania. Najbližšie k problematike je norma DIN 66 285 (od mája 1995 sa označuje ako DIN ISO/IEC 12 119) „Softvérové produkty. Požiadavky na kvalitu a testovanie“. Pritom je jasné, že nestačí posudzovať iba kvalitu výsledného produktu, ale aj kvalitu jednotlivých komponentov, ako napríklad bázu dát, procesy, moduly atď.

V literatúre [2, 4] sú uvedené nasledovné zásady, ktoré musia byť dodržané pri procese testovania:

- Zamerať sa na verifikáciu a validáciu všetkých fáz vývoja, t. j. koncepciu, požiadavky, riešenie, implementáciu, testy, inštaláciu, prevádzku a údržbu.
- Štandardizovať postupy (umožňujú redukovať námahu pri písaní a protokolovaní).
- Zaručiť opakovateľnosť testovania.
- Vylúčiť subjektívne vplyvy.
- Vypracovať detailnú skúšobnú dokumentáciu (ktorá umožňuje skúšku opakovať).
- Časovo usporiadať postup (predbežná kontrola, skúšanie, skúšanie zmenených častí).
- Logicky usporiadať postup (zoskupenie podobných funkcií a ich kontrolu, kontrolu jednotlivých samostatných častí systému, interakciu medzi subsystémami, celkovú kontrolu).
- Hierarchické usporiadanie postupu (týka sa dokumentácie).

Iní autori [3, 6, 7] pri testovaní rozlišujú nasledovné úlohy:

- Nutnosť vytvoriť obsah skúšok tak, aby sa preskúšali všetky prípady opisované v dokumentácii. Preveruje sa nielen informačný systém, ale povinne všetky podklady na strane skúšajúceho, t.j. predpoklady skúšky, inštalácia, opis produktu, dokumentácia, program, neprotirečivosť dokumentácia/program, neprotirečivosť produktu/dokumentácia, atď.

- Zhodnotiť obsah skúšok (ohľadom komplexnosti, náchylnosti k chybám, frekvencii výskytu a dôležitosti).
- Celkovo zhodnotiť a zoskupiť obsahy skúšok.
- Určiť hĺbku skúšania (povinné skúšky a polovica skupín s najväčšou dôležitosťou sa označí ako majoritná a testuje sa s veľkou hĺbkou, ostatné skupiny sa označia ako minoritné a netestujú sa hĺbkovo).
- Zadať obsahy skúšok – prideliť jednotlivé skúšky skúšajúcim.
- Skontrolovať predpoklady skúšky a inštalácie. Skontroluje sa, či je produkt nainštalovaný (na minimálnej hardvérovej konfigurácii) a či sú k dispozícii všetky potrebné podklady.
- Kontrolovať opis produktu.
- Vyhotoviť skúšobnú správu.

Testovanie na spoľahlivosť

Spoľahlivosť SW sa dá definovať ako pravdepodobnosť, že daný SW bude správne fungovať v danom prostredí počas daného časového cyklu. Na základe otestovania SW systému množinou testov sa dá matematicky vyjadriť spoľahlivosť, ktorú je možné od tohto systému očakávať, nasledovne (2):

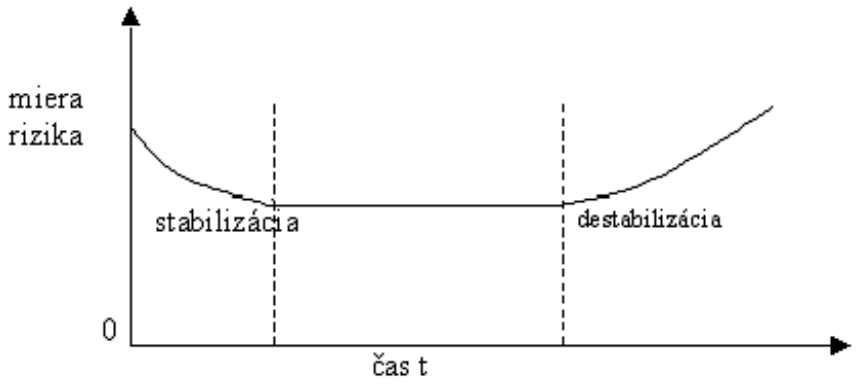
$$R = 1 - \frac{f}{n} \ ,$$

kde:
R je predpokladaná spoľahlivosť, n je počet spustených testov, f je počet neúspešných testov

Cieľom je, aby sa premenná R blížila k hodnote 1.

Keď je softvér vložený do väčšieho systému, je nevyhnutné zistiť, ako softvér prispieva k spoľahlivosti celého systému. Spoľahlivosť je základná štatistická miera kvality, vyjadrujúca pravdepodobnosť, že výrobok zlyhá vo svojom operačnom prostredí, v rámci danej fázy operácie.

Proces zlyhania softvéru je úplne odlišný proces, ako náhodný fyzický fenomén, ktorý je základom štatistického ošetrenia fyzických systémov. Všetky softvérové zlyhania sú výsledkom explicitných návrhových chýb. Ak je program spustený na chybných vstupných údajoch, dôjde k permanentnému zlyhávaniu; ak na správnych vstupoch, tak nikdy nedôjde k jeho zlyhaniu. Teória o softvérovej spoľahlivosti bola navrhnutá k analógii mechanickej spoľahlivosti.



Obr. 1. *„Vaňová krivka“ vyjadrujúca vzťah miery rizika v závislosti na čase*

Obr. 1 hovorí, že keď je fyzický systém nový, má väčšiu pravdepodobnosť zlyhať z dôvodu vzniku určitých nedostatkov. Potom sa "stabilizuje" a intenzita zlyhania sa zníži a zostane skoro konštantná. Napokon blízko konca jeho užitočného života sa systém stále viac blíži k pravdepodobnejšiemu zlyhaniu.

Niektoré programy pracujúce v režime nepretržitých operácií produkujú údaje zlyhania a sú prirodzene prezentované ako výsledná postupnosť.

Zo zaznamenaných časov zlyhania t_1, t_2,…, t_n začínajúcich na 0, je možné vypočítať priemerný čas do zlyhania (MTTF).

$$\text{MTTF} = \frac{t_1 + \sum_{i=1}^{n-1} (t_{i+1} - t_i)}{n} .$$

MTTF je vlastne primárny štatistický parameter kvality pre takéto programy. Je to predpoklad štatistických teórií pre nepretržite bežiace programy, ktorých vstupy, ktoré riadia vykonávanie programu sú "typické" pre svoje použitie [2].

Náhodné testovanie

Náhodné testovanie pripúšťa, že testovacia súprava je vzorka vzatá z miesta vstupu programu, pričom sa vyžaduje túto vzorku chápať ako nesystematickú chybu. Náhodné testovanie je užitočné na intuitívne predpovedanie chýb. Ak je použitá vhodná vzorka na testovaciu súpravu, výsledky na tejto vzorke obstoja v ďalšom správaní sa programu, ktorého vstupy sú neznáme.

Náhodné testovanie nemôže byť použité dovtedy, kým neexistujú prostriedky vytvárania vstupov "náhodne". Pseudonáhodné čísla z jednotnej distribúcie môžu byť použité ako testové vstupy, ak je známy rozsah vstupných hodnôt programu. V skutočných aplikáciách je tento rozsah daný hardvérovými limitmi. Napríklad matematická programová knižnica môže mať primeranú presnosť iba v určitom rozsahu, ktorý je daný jej špecifikáciou .

Štatistické predpovede nemajú opodstatnenosť dovtedy, kým vzorka nie je "reprezentatívna", čo pre softvér znamená, že testovacia sada musí byť navrhovaná obdobne.

Je daná funkcia *d*, ktorá sa nazýva hustota pravdepodobnosti, potom distribučná funkcia *F(x)* je kumulovaná pravdepodobnosť, ktorá je definovaná nasledovne [2]:

$$F(x) = \int_{-\infty}^{x} d(z)dz .$$

Pri generovaní testovacej sady podľa distribučnej funkcie F sa začína s generovaním pseudonáhodných reálnych čísel r rovnomerne rozložených na intervale (0,1) a generuje sa

$$F^{-1}(r) .$$

Hustota pravdepodobnosti *d* môže byť technicky daná ako časť špecifikácie programu a je nazývaná prevádzkový profil .

Metriky

Hoci bola schválená hlavná sada štandardov, príslušné metriky testovania či softvér vyhovuje štandardom nie sú stále dostatočne definované. Publikácie od IEEE (roky 1988 až 1998) prezentujú potenciálne metriky, ktoré môžu byť požívané na testovanie ľubovoľného atribútu. Problém je v tom, že ani jedna metrika nie je schopná jednoznačne odmerať čiastkové atribúty. Odlišné metriky môžu dať rozličné výsledky zoradenia toho istého atribútu, tvorba konkrétnych porovnaní naprieč produktom je ťažká a nespoľahlivá.

Nedostatok metrík kvality vedie väčšinu firiem iba k spočítaniu počtu chýb, ktoré sa objavili v priebehu testu. Niektoré organizácie sa pokúšajú využívať zdokonalené testovacie techniky, ako napríklad predvídanie poľa spoľahlivosti založeného na testovaných dátach a prepočítanej hustote chýb na skúšku kvality vo svojich produktoch [2, 3].

Keď sa navrhne nová metrika, objaví sa niekoľko charakteristík, ktoré sú spoločné s existujúcimi metrikami. Metriky prevažne používame pri vývoji softvéru, ale tieto nie sú výhradne iba pre softvér. Je možné pomocou nich testovať aj iné produkty.

Základné charakteristiky, ktoré by mali mať všetky metriky:

- Jednoduchosť a vypočítateľnosť: určenie metrík a používanie metrík je priamočiara a jednoduchá úloha.
- Presvedčivosť: metriky sa javia ako „merače“ správnych atribútov, zobrazujú platnosť.
- Trvalosť a objektívnosť: výsledky sú reprodukovateľné.
- Stálosť v častiach a rozmeroch: časti majú byť interpretovateľné a jasné.
- Nezávislosť programovacieho jazyka: metriky nemajú byť založené na špecifických úlohách a majú byť založené na typoch produktov, ktoré sú testované.
- Spätná väzba: výsledky metrík poskytujú užitočné informácie späť osobám vykonávajúcim test.

Pri kvalite metrík je dôležitý ten faktor, že určité softvérové atribúty sú viac vhodné na meranie ako iné atribúty. Metriky, ktoré sú ľahšie odmerateľné, sú taktiež prinajmenšom dôležité v eliminovaní nestálosti spotrebiteľského pohľadu na softvérovú kvalitu. Súčinnosť, spoľahlivosť a udržiavateľnosť sa ťažko merajú, lebo sú dôležité pri odhadovaní celkovej kvality softvérového produktu. Neschopnosť poskytnúť spoľahlivé a objektívne metriky pre niekoľko veľmi dôležitých atribútov je zjavné zlyhanie softvérovej metriky.

Metriky sú klasifikované do dvoch typov, a to či sú prediktívne alebo deskriptívne. Prediktívna metrika využíva predikciu o softvéri neskôr v životnom cykle návrhu. Je založená na predikovaní udržiavateľnosti softvérového produktu v chode.

Deskriptívna metrika opisuje stav softvéru v čase merania, t. j. metrika spoľahlivosti môže byť založená na počte systémových zlyhaní počas danej periódy.

Rôzni autori majú rôzne prístupy k metrikám. Dvaja zakladatelia hierarchických modelov, McCall a Boehm si osvojili tiež rôzne prístupy. Boehm zastával metódy založené na kontrolnom zozname, ktorý vyžadoval odpovede „áno/nie“. McCallov prístup je viac kvantitatívny, využíva výsledok odvodený z rovníc ako napr.:

$$\text{McCallov koeficient} = \frac{n_{01}}{n_{tot}},$$

kde n_{01} je počet modulov obsahujúcich iba výstupné body nula alebo jedna a n_{tot} je celkový počet modulov.

Vo všeobecnosti v tomto prístupe výsledky sú normalizované v rozsahu medzi 0 a 1, pre jednoduchšie kombinovanie a porovnanie.

Dôsledky nedostatočného testovania

V súčasnosti je dostupných množstvo výkonných metrík, procedúr a nástrojov na podporu testovania. Pri ich využívaní by náklady na certifikované testovanie mali poklesnúť a kvalita softvéru by sa mala zvýšiť, čo nie vždy je pravdou.

Dôsledky nedostatočného testovania môžeme zhrnúť do troch základných kategórií [3]:

Chyby spôsobené nízkou kvalitou

Najnepríjemnejším efektom nedostatočnej testovacej technológie je nárast zlyhaní a chýb produktov, ktoré sa objavili potom, ako bol produkt testovaný.

Ako je uvedené v tabuľke 2 na príklade kozmického a leteckého priemyslu, v dôsledku problematického softvéru boli stratené miliardy dolárov. Veľké chyby majú sklon byť viditeľné nielen pre odborníkov, ale aj pre laikov a výsledok je strata reputácie a ďalších obchodov pre spoločnosť.

ŠKODY SPÔSOBENÉ NEDOSTATOČNÝM TESTOVANÍM SOFTVÉRU Tabuľka 2

	Airbus A320	**Ariane 5 Galileo Poseidon Let 965**	**Pathfinder Lewis USAF Step**	**Zenit 2 Delta 3 Near**	**DS-1 Orion 3 Galileo Titan 4B**
	1993	1996	1997	1998	1999
Agregované náklady	nedostupné	640 mil $	116 mil $	255 mil $	1600 mil $
Straty na životoch	3	160	-	-	-
Strata dát	-	Áno	Áno	Áno	Áno
V tejto tabuľke nie sú zarátané straty, ktoré sa vyskytli v misii na Mars v roku 2000 a 2001 z dôvodu zlyhania softvéru.					

Zdroj: NASA IV&V Center, Fairmount, West Virginia.

Softvérové chyby sú klasifikované typom chyby, miestom vzniku, časom vzniku, úrovňou vážnosti, frekvenciou vznikania a nákladmi. Individuálne chyby potom môžu byť agregované podľa nasledujúceho prístupu:

- **Nedostatočné podriadenie sa štandardom** – problémom je, že softvérové funkcie, dátové reprezentácie, preklady alebo interpretácie neboli prispôsobené procedurálnym procesom alebo formátu, ktorý špecifikuje štandard.
- **Nedostatočná prevádzkyschopnosť s inými produktmi** – problémom je neschopnosť softvérových produktov vymieňať a zdieľať dáta s inými systémami.
- **Slabá výkonnosť** – problémom je, že aplikácie síce pracujú, ale nie tak ako by sa očakávalo a ako je žiadané.

Nárast nákladov na vývoj softvéru

Historicky je dané, že proces identifikovania a odstránenia chýb v procese vývoja softvéru reprezentuje viac ako polovicu nákladov na celý vývoj. V závislosti od použitej metódy testovacie aktivity zaberajú od 30 do 90 percent práce vydanej na vytvorenie funkčného programu. Včasné odhalenie chyby môže výrazne redukovať náklady. Chyby môžu byť klasifikované na základe toho, kde boli objavené, to znamená v jednotlivých etapách životného cyklu vývoja softvérového produktu. Konkrétne sa jedná o etapy požiadavky, návrh, kódovanie, testovanie modulov, integračné testovanie, systémové testovanie, inštalačné a akceptačné testovane a prevádzkovanie a údržba. V tab. 3 je vidieť, že čím neskôr objavíme chybu v procese vývoja, tým bude nákladnejšie ju odstrániť. Ďalším dôležitým faktorom, ktorý je vidieť, je obrovský nárast nákladov na odstránenie chyby v priebehu dvadsiatich rokov. Najvypuklejšie sa to týka finálnych etáp vývoja softvéru.

NÁKLADY NA ODSTRÁNENIE CHYBY V ZÁVISLOSTI NA ETAPE ŽIVOTNÉHO CYKLU Tabuľka 3

Etapa životného cyklu	**Náklady na odstránenie chyby [x- je jednotka nákladov vyjadrená v človekohodinách alebo peniazoch]**	
	Štúdia p. Baziuka - 1995	**Štúdia p. Boehma - 1976**
Požiadavky	**1x**	**0,2x**
Dizajn		**0,5x**
Kódovanie		**1,2x**
Testovanie modulov		
Systémové testovanie	**90x**	**5x**
Inštalačné testovane	**90x – 440x**	**15x**
Akceptačné testovane	**440x**	
Prevádzkovanie a údržba	**470x - 2900x**	

Zvýšený čas na umiestnenie softvéru na trh

Nedostatok štandardizovaných technológií testovania má tiež vplyv na nárast času, ktorý je potrebný na umiestnenie produktu na trh. To úzko súvisí s potenciálnou stratou príležitosti na dobrý výsledok predajnosti a tým aj návratnosti nákladov na vytvorenie produktu. Strata príležitosti na dobrý výsledok predajnosti má za následok podobný efekt, ako objavenie chyby už v distribuovanom produkte. Ak by boli ľahko dostupné štandardizované testovacie procedúry, zamestnanci poverení testovaním by strávili menej času vývojom vlastných testovacích technológií.

Štandardizované testovacie technológie by mali urýchliť vývoj tým, že sa zníži potreba:

- vyvíjať špecifický testovací softvér pre každú aplikáciu,
- vyvíjať špecifické testovacie dáta pre každú aplikáciu,
- používať metódu „pokus a omyl“ pri určení, ako používať neštandardné testovacie nástroje.

Na záver uvedieme nasledovnú zaujímavú informáciu prezentovanú v roku 2002 Shaferom (uvedené čísla sa vzťahujú na celý svet).

Pokiaľ by až/iba **99,90%** všetkých funkcií v produkte bolo správnych, tak by:

- 9 703 bankových transakcií bolo každú hodinu preplatených z iných bankových účtov.
- 27 800 dopisov bolo stratených každú hodinu.
- 8 605 komerčných letov havarovalo každoročne v priebehu štartu.

Zoznam bibliografických odkazov:

[1] PATON, R. *Testováni softwaru.* Computer press, 2002.
[2] GILLIES, A. *Software Quality – Theory and management*. Chapman and Hall, 1998.
[3] US Department of Commerce Technology Administration: The Economic Impact of Inadequate Infrastructure for Software Testing. 2002.
[4] FEWSTER M., GRAHAM, D. *Software Test Automation*. McGraw-Hill, 1999.
[5] TANUŠKA P., SCHREIBER P. Poznámky k problematike testovania informačných systémov. In *ATP Journal*, 1999, 10. ISSN 1335-2237
[6] MUDRONČÍK, D. Validation of process control systems based on GAMP. In *14th conference on Process Control*, 2003.
[7] TickIT Guide: Using ISO 90001:2000. British Standards Institution. 2001.