ROZDIELNOSŤ PRÍSTUPU K TESTOVANIU V TRADIČNÝCH A AGILNÝCH METODIKÁCH

„Skúsili ste to vypnúť a znovu zapnúť?“

Bc. Ján Kováč

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jan.kovac[zavináč]windowslive[.]com

Abstrakt. *Keďže výsledkom procesu vývoja softvéru je vytvorenie určitého výrobku, nastáva otázka ako zabezpečiť jeho kvalitu. Avšak na rozdiel od iných výrobkov je kvalita softvéru ťažšie merateľná a taktiež aj ťažšie definovateľná. Nedokážeme konkrétne povedať čo je kvalitný softvér. Na kvalitu sa môžeme pozerať z viacerých uhlov pohľadu – z pohľadu programátorov, projektového manažéra, koncových používateľov atď. V dnešnej dobe je čoraz viacej firiem poskytujúcich softvérové riešenia a tak je kvalita jedným z faktorov, ktoré ovplyvňujú ich postavenie na trhu a tvoria dobré meno firmy. Jednou z metód zabezpečenia kvality softvéru je aj testovanie. V tejto eseji opíšem niektoré problémy testovania, ale hlavne porovnám rôzne prístupy k testovaniu v niektorých používaných metodikách, keďže nie každá metodika sa stavia k testovaniu rovnako. A taktiež opíšem princípy testovania využívané v agilných metodikách vývoja SW a porovnám ich s tradičným prístupom.*

Kľúčové slová: *testovanie, kvalita, verifikácia, validácia*

Kvalitný softvér?

Ak chceme zistiť čo je to kvalitný softvér a spýtame sa viacerých skupín ľudí, počnúc používateľmi cez vývojárov až po manažment, čo by mal spĺňať softvér, ktorý by bol v ich ponímaní kvalitný, zistíme, že každý má inú predstavu ako by mal vyzerať a ako by mal fungovať. Budúci používatelia nám zrejme povedia, že softvér by pre nich mal byť jednoducho použiteľný, mal by fungovať vždy, v každom čase keď ho potrebujú použiť,

mal by im poskytovať intuitívne ovládanie a mal by robiť presne to, čo od neho očakávajú aby robil. Ak sa spýtame začínajúceho programátora na jeho definíciu kvalitného softvéru odpoveď môže byť - softvér, ktorý „nepadá“. Programátori starajúci sa o rozširovanie možností daného softvéru, úpravu zdrojových kódov, refaktoring atď., môžu za kvalitný softvér pokladať taký, ktorého zdrojový kód je ľahko pochopiteľný, rozšíriteľný a udržiavateľný. Ja som sa taktiež pokúsil od ľudí vo svojom okolí zistiť ich predstavu o kvalitnom SW a dominovali najmä tieto dve odpovede – jednoduché použitie a aby program fungoval vždy, keď ho je potrebné použiť. Jednou z požiadaviek na kvalitu býva aj bezpečnosť. Tá vzrastá pri softvéri, ktorý môže zapríčiniť smrť, vysoké finančné škody, ohrozenie životného prostredia a pod.

Tu môžeme vidieť, že kvalita softvéru je definovaná množinou viacerých atribútov. Ak rozdelíme atribúty na tie, ktoré očakávajú ľudia, ktorí sú nejakým spôsobom zapojený do procesu vývoja daného softvéru t.j. vývojári, manažment, testeri a na tie, ktoré má zadávateľ projektu resp. koncoví používatelia, môžeme vlastnosti výsledného produktu rozdeliť na vlastnosti, ktoré pre používateľov viditeľné sú a na také, ktoré pre nich viditeľné nie sú. O viditeľných vlastnostiach sa dá hovoriť v súvislosti s používateľským rozhraním aplikácie: ako vyzerajú dialógy? Ako prívetivo program komunikuje?

Softvér je tvorený podľa toho, čo si objedná zákazník. Z toho nám vyplýva že aj program, ktorý je z pohľadu vývojárov zlý, zbytočný alebo nepotrebný môže byť zákazníkom každodenne využívaný a tak hodnotený kladne a považovaný jeho používateľmi za kvalitný. Ako vývojári by sme sa mali snažiť v prvom rade uspokojiť požiadavky zákazníka.

Niet pochýb o tom, že každý vývojársky tím sa snaží dodať na trh čo najkvalitnejší produkt. Ale čím je projekt väčší, tým viac v ňom nájdeme chýb. V súčasnosti je veľmi diskutovanou témou testovanie. Testovanie nezabezpečuje priamo kvalitu SW ale dopomáha k tomu, aby vyvíjaný softvér nemal chyby, ktoré sú pre používateľov znakom „nekvality“.

Príchodom agilných metodík sa ale zmenili aj niektoré prístupy k testovaniu. Testovaniu je prikladaný ešte väčší význam ako doposiaľ. V čom sa tento prístup líši a je vôbec lepší? Hodia sa agilné metodiky, čo sa týka testovania, na každý typ projektu? Aj na tieto otázky sa pokúsim zodpovedať.

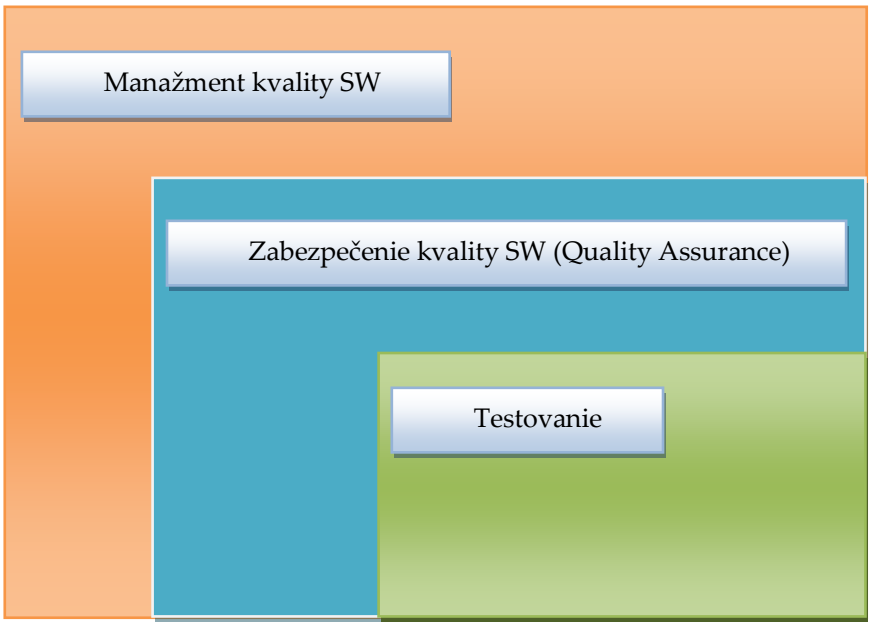
Zabezpečenie kvality SW a jeho vzťah k testovaniu

Ale vráťme sa ešte raz na začiatok a vysvetlime si proces zabezpečenia kvality SW a jeho súvislosť s procesom testovania. Proces zabezpečenia kvality je označovaný ako Software Quality Assurance (QA), jeho cieľom je zabezpečenie požadovanej funkcionality, pokiaľ možno bez chýb a taktiež zabezpečenie toho, aby bol projekt vykonávaný v súlade s danou špecifikáciou.

O zabezpečenie kvality by sme sa mali snažiť počas celého vývojového cyklu. Jedna možnosť ako toto splniť je používanie metodík, pričom nie každá metodika je vhodná pre každý typ projektu. Okrem toho existuje viacero metód, ktoré nám pomáhajú zabezpečiť kvalitu pre už spomínané atribúty opisujúce kvalitu produktu.

Jednou z metód zabezpečenia kvality je aj testovanie. Testovanie je odskúšanie funkčnosti produktu na testovacej vzorke dát, ktoré sa môžu vyskytnúť pri použití

v reálnych podmienkach. Do oblasti testovania spadajú aj pojmy verifikácia a validácia, ktoré si vysvetlíme neskôr. Avšak nemali by sme si myslieť, že testovanie nám zabezpečí kvalitu. Veľa ľudí si myslí, že ak chcú vyvinúť kvalitný softvér, stačí im testovať. Dalo by sa povedať, že testovanie je podmnožinou zabezpečenia kvality (QA) ako to môžeme vidieť na Obr. 1.



Obr. 1. Vzťah medzi testovaním a zabezpečením kvality.

Procesy testovania a zabezpečenia kvality sa prekrývajú, ale podľa ich primárnych úloh ich môžeme odlíšiť:

- hlavnou úlohou softvérového testera je vyhľadávať chyby; vyhľadať ich čo najskôr a zabezpečiť ich nápravu
- hlavnou úlohou osoby, zodpovednej za zabezpečovanie kvality SW, je vytvorenie a presadenie vhodných štandardov a metód, ktoré zdokonalia proces vývoja a zabránia vo vzniku chýb

Keďže je testovanie často skloňované a diskutované, možno by sa zdalo že patrí medzi najúčinnejšie metódy zabezpečenia kvality. Opak je pravdou. Percento odhalenia chýb v testovaní nie je príliš vysoké. Účinnosť v hľadaní chýb je 25-45%, zatiaľ čo účinnosť niektorých ďalších metód je oveľa vyššia. Napríklad formálne Faganovské inšpekcie majú účinnosť až 60% a prototypovanie môže dokonca odhaliť až 70% chýb [2].

Testovanie má za úlohu nájsť chyby v SW. Bohužiaľ nič nie je dokonalé a nie vždy sa tieto lokalizované chyby skutočne aj opravia.

Na tomto mieste by sme si mali objasniť čo všetko by sme mohli považovať za chybu. Definície podľa Pattona [4]:

- SW robí niečo, čo nie je spomenuté v špecifikácii

- SW robí niečo, čo by podľa špecifikácie robiť nemal
- SW robí niečo, čo nie je spomenuté v špecifikácii, ale malo by byť

Tento zoznam ale samozrejme nie je kompletný. Je dosť problematické vymenovať všetky stavy, ktoré sú interpretované ako chyby. Budúci používatelia môžu považovať za chybu aj stav, keď je daný softvér „pomalý".

Verifikácia vs. validácia

V procese testovania sme sa mohli stretnúť s dvomi dôležitými pojmami – verifikáciou a validáciou. Ale vieme skutočne čo je verifikácia a čo validácia? Tieto pojmy bývajú často nesprávne interpretované alebo zamieňané. Rozdiely medzi nimi sú niekedy zle pochopené aj z toho dôvodu, že sú v rôznych vedeckých časopisoch alebo odbornej literatúre zaoberajúcej sa testovaním, resp. všeobecne vývojom softvéru definované rozlične. Z tohto dôvodu sa pokúsim o jednoduchú definíciu týchto pojmov. Verifikácia je proces, ktorý slúži na overenie toho, či je softvérový produkt vyvíjaný správne. Výstupom procesu verifikácie je stav, kedy môžeme povedať, či sme splnili zadanú špecifikáciu. Ale musíme si uvedomiť, že verifikácia nie je postačujúcou podmienkou – príklad kompilátor. Postačujúcou podmienkou nie je preto, lebo aj samotná špecifikácia môže byť chybná. Validácia nám oproti tomu má povedať, či je vytváraný produkt správny, tzn. či slúži používateľovi tak ako si to on predstavuje. Validácia na rozdiel od verifikácie nám overuje produkt vzhľadom na reálne požiadavky. Validácia je to, čo je v konečnom dôsledku potrebné.

Nikto nie je dokonalý – ani testovanie

Testovanie má určite veľa nedostatkov o ktorých už bolo napísaných veľa kníh a veľa ďalších sa napíše. Všetky sa nedajú spomenúť ale aspoň tie najviac diskutované by som rád spomenul. Ako už bolo spomínané v úvode, testovanie nám neodhalí resp. neodstráni všetky chyby vo vyvíjanom produkte. Testovanie naráža na viaceré problémy ako napr.:

- Výsledky testovania môžu byť ovplyvnené buď zlou komunikáciou prebiehajúcou medzi zákazníkom a tímom alebo aj v rámci samotného tímu.
- Testovanie býva podceňované.
- Testovanie je citlivé na vstupné údaje (problém zohnať realistickú vzorku údajov v potrebnom množstve).
- Chýbajúca resp. nejednoznačná špecifikácia.
- Produkt sa nedá otestovať celý.
- Testy je nutné neustále prispôsobovať zmenám. Aj malá zmena v SW môže spôsobiť, že testy nebudú pracovať správne.

Ako je možné, že testovanie býva podceňované keď je kvalita považovaná za takú dôležitú? Môže za to určitá „nepopularita" testovania u vývojárov. Súvisí to s viacerými skutočnosťami medzi ktoré môžeme zaradiť najmä tieto:

- Ľudia (vývojári) radšej vytvárajú „niečo nové“ ako keby mali opravovať „pokazené“.
- Testovanie býva v procese vývoja SW na poslednom mieste t.j. pred dodaním produktu zákazníkovi. Ale zdržaný projekt môže spôsobiť, že na testovanie nebude dosť času. Následne sa môže spŕška kritiky zniesť na testerov.
- Testovanie nie je nikdy konečný proces.

Testovanie v tradičných metodikách

Ešte v dobách keď neboli známe žiadne metodiky vývoja softvéru, bol softvér písaný štýlom Napíš a oprav (Code and Fix). Programátori vtedy väčšinou pre seba programovali program, ktorý by im mal pomôcť v práci. Bolo to bez akýchkoľvek príprav, sadli k PC a programovali. Následne, ak sa počas práce s programom prišlo na niektoré chyby, chyby sa hľadali a opravovali a tak to šlo stále dokola. Pre vtedajšie podmienky a malé programy (čo do rozsiahlosti) to stačilo. Tento prístup ale viedol k známej softvérovej kríze. A to bol dôvod vzniku metodík. Metodiky vzniknuté po softvérovej kríze kládli silný dôraz na fázy analýz, špecifikácií, testov apod. Tieto metodiky t.j. neagilné boli vyvinuté skôr a nepočítali so zmenou špecifikácie počas vývojového cyklu. Avšak ešte donedávna tieto metodiky resp. modely životného cyklu SW, ako napr. vodopád, ktorý je typickým predstaviteľom tejto skupiny metodík, stačili vývojárom v ich práci. Dnes význam týchto metodík klesá a firmy poskytujúce softvérové riešenia stále vo väčšej miere prechádzajú na agilné metodiky. Ale aj napriek tomu si stále vo veľkom množstve firiem držia svoje neohrozené postavenie.

Najväčšou nevýhodou týchto metodík je, že celý vývojový cyklus prebieha sekvenčne. Až po dokončení jednej fázy sa pokračuje vykonávaním ďalšej. Testovanie prichádza na rad až po vlastnej implementácii. Myslím, že tento prístup je nevýhodný a je to len „čakanie na chyby“ a ich následné hľadanie a opravenie. A taktiež ďalšia nevýhoda plynúca z umiestnenia fázy testovania až po implementácii je v tom, že ak je projekt zdržaný (a to sa stáva až príliš často), nie vždy zostane čas na dôkladné testovanie. V najhorších prípadoch, keby nezostal žiadny čas na testovanie, otestoval by produkt až budúci používateľ (zákazník). Nie je asi nutné dodávať, že tak môžu vzniknúť veľmi nepríjemné situácie.

Nevýhoda tohto testovania, ktoré prichádza na rad v dosť neskorej fáze je aj zvýšenie nákladov. Ak by sme boli schopní nájsť skôr chyby, bola by rýchlejšia ich oprava a taktiež by to znamenalo ušetrenie nákladov. A to je to, na čo mysleli ľudia stojaci v pozadí vzniku agilných metód.

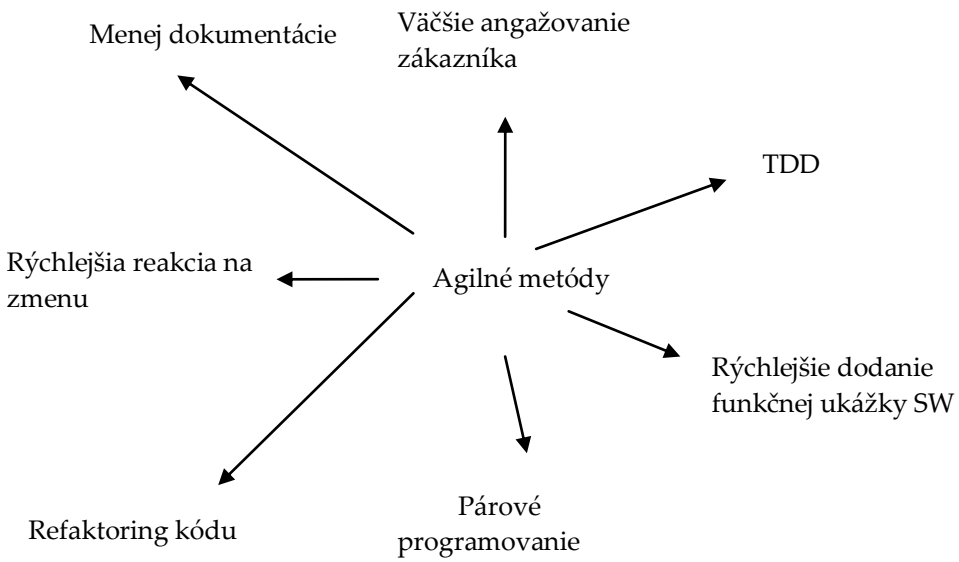
Rola softvérového testera v tradičných metodikách vývoja softvéru by sa dala charakterizovať približne takto :

- Nájdenie chýb vo vyvíjanom produkte, ako napr. chyby v požiadavkách
- Dokázanie, že softvér spĺňa tieto požiadavky

Testovanie v agilných metodikách

Na začiatok by som ich aspoň v skratke zhrnul. Agilné metodiky patria medzi novšie druhy metodík. Boli vytvorené ako odpoveď na problémy známeho V-Modelu testovania a už spomenutého vodopádového modelu životného cyklu, týkajúce sa častých zmien požiadaviek, ktoré sa neustále menia. Tým pádom to, čo je aktuálne vyvinuté nemusí korešpondovať s tým, čo chce alebo čo potrebuje momentálne zákazník. Ako vývojári môžeme reagovať pružnejšie na tieto podnety. Aj to je jeden z dôvodov prečo sú agilné metódy stále viac nasadzované.

Ale ako sa líši prístup k testovaniu v agilných metodikách? Stratégia testovania a prístup agilného vývoja môžu byť veľmi odlišné od tradičných „byrokratických" metodík. Agilné metodiky vychádzajú z princípu, že pri písaní kódu môžu vznikať chyby. Dôraz je kladený na čo najskoršie opravenie nájdených chýb. To je rozdiel oproti tradičným metodikám (napr. vodopád), kde je tvorba testov naplánovaná až po samotnej implementácii. Ďalšie špecifiká agilného vývoja ukazuje Obr. 2.



Obr. 2. Špecifiká agilných metód [3].

Faktory ovplyvňujúce kvalitu ako napr. párové programovanie sú z môjho pohľadu výhodné v tom, že vzniká menej chýb a tak je v programe obsiahnuté menšie množstvo chýb. Nevýhoda tkvie vo väčších nákladoch na pracovnú silu.

Metodika najviac zameraná na testovanie je TDD, ktorú by som rád opísal konkrétnejšie.

Testami riadený vývoj softvéru

Testami riadený vývoj (Test Driven Development - TDD) považuje práve testovanie za najdôležitejšiu časť vývoja. Práve preto som ho vybral ako vzorovú metodiku

reprezentujúcu agilné metódy vývoja. Dokonca testovanie je v nej dôležitejšie ako samotné písanie kódu. Vytváranie testov sa uskutočňuje ešte pred napísaním kódu. Tento prístup podľa mňa prináša radu výhod, ako napr. otestovanie každej časti kódu. Takže na rozdiel napr. od vodopádového modelu nečakáme na chybu až kým sa stane ale snažíme sa vytvoriť len malé množstvo kódu neobsahujúce veľa chýb. Idea je to dobrá ale nie vždy sa to dá zaručiť. Za prípravu testu ešte pred napísaním kódu je zodpovedný programátor. Programátor ale má vždy tendenciu programovať a tak sa môže stať, že zabudne resp. vôbec žiadne testy neurobí. Práve z tohto dôvodu je nutná určitá disciplína a odhodlanosť pri používaní TDD [0].Okrem toho musíme vykonávať v TDD aj iné druhy testovania, ako napríklad testovanie používateľského rozhrania, akceptačné testovanie, testovanie integrácie atď.

Testy v TDD nahrádzajú aj písanie dokumentácie a špecifikácie, keďže test sám o sebe poskytuje návod ako používať konkrétnu časť kódu, a tým ušetríme čas čítaním dokumentov. Neznamená to ale, že dokumentovať v TDD nemusíme.

Záver

Testovanie je neodmysliteľnou súčasťou každého cyklu vývoja softvéru. Avšak testovanie ako také nedokáže úplne zabezpečiť odstránenie chýb v softvérových produktoch. Ale nie je to len kvôli rozsiahlosti systémov, ale aj kvôli hore uvedeným príčinám. Jedna z týchto príčin je v ľuďoch angažovaných vo vývoji softvéru. Testovaniu prikladajú nedostatočnú dôležitosť, prípadne manažment snažiaci sa o rýchle dodanie softvéru skracuje čas potrebný na testovanie aby sa nezvyšovali náklady.

Ak si vezmeme, či už tradičné alebo agilné metódy, jasne môžeme vidieť rozdielne prístupy a spôsoby ktorými sa stavajú k testovaniu. Ak si tieto dva prístupy porovnáme, vidím že agilné metodiky prikladajú testovaniu väčšiu dôležitosť. Testovanie v nich môže veľmi kladne ovplyvniť aj kvalitu výsledného produktu. Aj napriek tomuto faktu softvér nevieme urobiť dokonalý. Prišiel som k názoru, že ako aj tradičné, tak aj agilné metodiky majú svoje výhody aj nevýhody. Agilné metodiky dokážu byť za určitých okolností výhodnejšie ale záleží to od „morálky“ vývojárov.

Použitá literatúra

1. Ambler, S. W. (2002). *Introduction to Test Driven Design (TDD)*. Dostupné na Internete: www.agiledata.org: http://www.agiledata.org/essays/tdd.html
2. Kadlec, V. (2004). *Agilní programování.* Brno: CPress.
3. Krishna, V. (2008). *Agile and Other Development Methods*. Dostupné na Internete: www.codeproject.com: http://www.codeproject.com/KB/architecture/Agile_Methods.aspx
4. Patton., R. (2002). *Testování softwaru.* Brno: CPress.

Annotation

Difference in access to testing in traditional and agile methodologies

Because result of software development process is to create a product, the question arises as to ensure its quality. However, unlike other products, software quality is measured and also heavier and more difficult to define. We can not specifically say what is good software. The quality can be seen from several angles - from the perspective coder, project manager, end users, etc.. Nowadays, more and more companies providing software solutions and so the quality is one of the factors that affect their market position and reputation are firm. One method of quality assurance and software testing is. In this essay I will describe some of the problems of testing but mainly compare different approaches to testing methodologies used by some as not every methodology is built on the same testing. And also describe the principles of testing used in agile software development methodologies and compare them with traditional approaches.