Je naozaj potrebný samostatný testovací tím?

MILOŠ RADOŠINSKÝ

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

Abstrakt. K tejto otázke ma „vyprovokovala" prax. Konkrétne moje krátkodobé pôsobenie v testovacom tíme. Môžete namietať, že krátkodobé, cca dvojmesačné pôsobenie ma neoprávňuje vyslovovať pochybnosti a dávať súdy. A možno práve naopak ma k tomu oprávňuje, keď už tak skoro vidím nedostatky. Tie v krátkosti zhrniem. Testovací tím je zvláštne označenie niečoho, čo sa tímom podľa mňa nedá nazvať. Veď tím by mal byť najmä o spolupráci, ktorá vedie k väčšiemu úžitku, ako je suma úžitkov práce jednotlivcov. V testovacom tíme toto nepozorujem. Výsledky jedného člena tímu totiž ostatní členovia nevyužívajú, a ak áno. len v obmedzenej miere. Každý člen tímu "hrabe" na svojom piesočku. Buď testuje manuálne, alebo píše automatické testy pre jemu pridelenú uzavretú množinu požiadaviek. Súdržnosť testovacieho tímu je teda malá. A to je problém. Keďže každý tester testuje svoju uzavretú množinu požiadaviek, nie je lepšie, aby bol v tíme vývojárov, ktorí tieto požiadavky implementujú? Nemal by byť bližšie k zdroju diania a možných zmien produkčného kódu? Každá minca má však dve strany. V tejto eseji sa preto pokúsim zhrnúť pre a proti samostatnému tímu.

Úvod

Čo je to testovací tím? Aké sú jeho kompetencie? Je to tím ľudí, ktorý pokrýva celý životný cyklus testovania od návrhu testov až po vykonávanie a vyhodnocovanie testov? Alebo len nejakú jeho časť?

Hneď na začiatku treba zdôrazniť, že testovanie vo všeobecnosti prebieha na rôznych úrovniach abstrakcie softvéru. Na každej úrovni pritom musí testovací tím dokonale poznať testovaný produkt a problematiku za účelom otestovania správnosti produktu alebo jeho časti. Napr. na úrovni unit testov musí dokonale poznať interný návrh modulu, aby napr. technikou bielej skrinky dokázal overiť pokrytie celého kódu modulu. Na úrovni integračných testov musí poznať architektúru systému alebo návrh komponenty za účelom overenia funkcionality na tejto úrovni. Všeobecne, na každej

úrovni musí testovací tím poznať špecifikáciu požiadaviek a návrh softvéru, aby vedel čo a ako testovať.

Vyššie uvedené môže intuitívne viesť k záveru. že testovací a produkčný tím by mali mať k sebe „blízko“. Čím bližšie tým lepšie, organizačne a fyzicky. Napr. unit testy by mal navrhnúť a realizovať samotný autor modulu, vývojár. Podobne na vyšších úrovniach. Napr. na úrovni komponentov autor komponenty je ten, kto najlepšie vie, aké moduly a rozhrania implementujú požadovanú a exportovanú funkcionalitu komponenty. Aké sú požadované vstupy a výstupy.

Avšak bez hlbších úvah a analýz nemožno prijať tento záver ako fakt. Bráni nám v tom niekoľko skutočností. Testovanie a overovanie kvality ako také nie je len o overovaní *správnosti*. Testovať sa dajú aj ďalšie atribúty kvality ako *použiteľnosť, spoľahlivosť, výkonnosť* a iné. Tieto atribúty sa overujú prevažne z pohľadu celého systému. V tomto prípade autor a realizátor testov nemusí poznať návrh systému alebo jeho časti.

Do hry pre samostatný testovací tím vstupuje aj ľudský faktor. Osoba nezainteresovaná do detailov návrhu, nezaťažená zaužívaným a jednostranným pohľadom dokáže odhaliť viac chýb ako autor danej úrovne. Tento praxou overený fakt možno demonštrovať aj touto analógiou. Autor článku po jeho vyhotovení kontroluje preklepy. Nedokáže ich však nájsť všetky. Niektoré chyby prehliadne, pretože text pozná už skoro naspamäť a číta ho správne, tak ako by mal byť napísaný, nie tak ako je napísaný naozaj.

V-model alebo ako to funguje bežne v praxi

V-model asi každý dôverne pozná. Je to asi prvý model vývoja softvéru, s ktorým sa stretol. Preto som ho zaradil hneď na úvod a budem sa naň v ďalšom texte intenzívne odvolávať ako na “vzorovú metódu“.

Podstatou V-modelu je sekvenčný prístup, rôzne úrovne testovania a skutočnosť, že pre každú etapu vývoja existuje zodpovedajúca etapa/úroveň testovania. Pritom ľavá vetva reprezentuje etapy vývoja a pravá vetva reprezentuje etapy testovania. Ide o abstraktný model, ktorý ďalej nešpecifikuje ďalšie detaily v jednotlivých etapách.

Existujú rôzne variácie V-modelu, ktoré sú prispôsobené špecifickým potrebám pre dané použitie. V-model možno použiť napr. ako súčasť iteratívneho a inkrementálneho vývoja, kde každú iteráciu možno chápať a plánovať ako V-model vývoja. V tomto prípade sa v každej iterácii špecifikuje, navrhne, vyrobí a otestuje inkrement použitím V-modelu.

V-model možno tiež použiť na vývoj systémov, kde sa požiadavky špecifikujú na rôznych úrovniach abstrakcie postupným zjemňovaním a dekompozíciou. V tomto prípade sa požiadavky na N+1 úrovni abstrakcie dekomponujú na požiadavky na N-tej úrovne. Napr. jedna požiadavka na subsystémovej úrovni môže byť dekomponovaná na niekoľko požiadaviek v rôznych komponentoch na komponentovej úrovni. Pričom subsystém je zložený z jednotlivých komponentov. Opačný proces prebieha následne

pri testovaní a integrácii. Tu sa testujú jednotlivé požiadavky na všetkých stupňoch abstrakcie od najnižšej až po najvyššiu.

Uvedené variácie možno použiť aj kombinovane a každá z nich už sama o sebe zvyšuje kvalitu softvéru zlepšením procesu vývoja oproti klasickému V-modelu. Prvá menovaná variácia zlepšuje kvalitu tým, že produkt je vytváraný postupne v malých dávkach, ktoré sa priebežne testujú. Produkt sa v každom novom zostavení/build-e opakovane testuje buď celý alebo regresne. Druhá menovaná variácia zlepšuje kvalitu možným zvýšením spektra granuality testov.

Proces testovania

Proces testovania sa realizuje za účelom dosiahnutia kvality softvéru. Samotný proces testovania môže byť podľa [5] rozdelený do troch fáz:

1. Plánovanie testov
2. Výber, realizácia a beh testov (testovací cyklus)
3. Vyhodnotenie

Plánovanie testov sa zaoberá základnými otázkami testovania: čo a ako treba testovať, kto a kedy bude testy realizovať. Testy treba teda najskôr identifikovať, navrhnúť a implementovať, aby sa mohli realizovať a následne vyhodnotiť. Plánovanie testov ale nie je jednoduchý proces. Pri plánovaní testov sa treba vysporiadať aj s týmito problémami [5]:

- Fázy testovania
- Stratégia testovania
- Zodpovednosti medzi skupinami
- Zmena požiadaviek
- Ľudské zdroje

Fázy testovania

Plánovanie fáz testovania vychádza z použitého vývojového modelu, ktorý sa pri tvorbe softvéru používa. Napr. v prípade V-modelu sú to jednotlivé úrovne testovania – testy jednotiek, integračné a systémové testy. Manažér testov tu rozhoduje o tom, aké fázy testovania a aké typy testov v jednotlivých fázach budú použité.

Jednotlivé fázy musia byť viditeľne oddelené na základe vstupno - výstupných podmienok. To znamená, že testovací tím musí presne vedieť kedy jedna fáza končí a kedy druhá začína. To platí najmä v prípade klasického V-modelu, kedy jednotlivé fázy testovania pokrývajú celý produkt, a teda sú na sebe vysoko závislé. Pri iteratívnom vývoji, keď jedna iterácia „vyrobí“ iba malý „kúsok“, striktné oddeľovanie fáz nie je až tak kritické. Ale aj v tomto prípade sa musí dodržať následnosť testov,

ktoré pokrývajú súvisiace požiadavky z viacerých úrovní abstrakcie (napr. subsystémové a komponentové požiadavky).

Stratégia testovania

Definovanie stratégie testovania úzko súvisí s definovaním fáz testovania. Stratégia testovania opisuje akým spôsobom sa bude softvér testovať v jednotlivých fázach. Ide o výber z veľkej palety typov testov: white box vs. black box testovanie, automatické vs. manuálne testovanie, dátovo orientované vs. krokovo orientované testovanie. Pre každú fázu testovania musí manažér testov vybrať vhodnú kombináciu vyššie uvedených typov testov.

V špecifických prípadoch manažér testov tiež rozhoduje o použitých nástrojoch pre automatizáciu testov. Pre štandardné vývojové technológie a behové prostredia (J2EE, Net) môže použiť štandardné nástroje ako napr. *JUnit, NUnit, Cactus*, *Spy*. Pre špecifické a neštandardné prípady musí vytvoriť vlastné.

Zodpovednosti medzi skupinami

Testovací tím nie je nezávislý. Práca testovacieho tímu závisí od výsledkov práce ostatných tímov/skupín ľudí – programátori, návrhári, manažéri, atď. Vzájomná zodpovednosť viacerých tímov identifikuje oblasť ich vzájomného pôsobenia a spolupráce. Napr. vzájomná zodpovednosť testovacieho a produkčného tímu môže byť spoločne konzultovať návrh testov za účelom ich automatizácie. Zodpovednosti medzi skupinami identifikujú a generujú úlohy, s ktorými treba počítať pri plánovaní. Najmä pri odhadovaní úsilia.

Keď majú dva tímy vzájomnú zodpovednosť za niečo, vzniká medzi nimi závislosť alebo zviazanosť. Hoci vysoká zviazanosť tímov sa "odsudzuje", dovolím si "kacírsku otázku". Čo je lepšie v tomto prípade, vysoká alebo nízka zviazanosť? Podľa mňa vysoká zviazanosť. Pretože testovací kód a iné testovacie artefakty sú vysoko závislé od produkčného (testovaného) kódu.

Položme si otázku ešte inak. Prečo by mali byť tímy (testovací a produkčný) zviazané minimálne, keď výsledky ich práce sú vysoko závislé, hoci len jednostranne – testovací kód závisí od produkčného kódu? Na uvedenú otázku v protiklade mojich očakávaní odpovedá prax. V praxi sa totiž často využíva na testovanie externý dodávateľ vo forme outsoursingu. Pričom produkčná firma pridelí z vlastných zdrojov iba ľudí, ktorí dozerajú na priebeh testov.

Zmena požiadaviek

Na zmenu požiadaviek alebo zmenu návrhu musí reagovať aj testovací tím v procese testovania. Aktuálna množina testov musí byť vždy v zhode s aktuálnou množinou požiadaviek. To sa dá docieliť jednoducho mapovaním požiadaviek na testovacie prípady a použitím nástroja na sledovanie zmien. Horšie je to v prípade zmeny návrhu. Niektoré zmeny v návrhu si totiž môžu vyžiadať aj komunikáciu a konzultáciu s produkčným tímom.

Ľudské zdroje

Ľudské zdroje podobne ako iné zdroje treba plánovať a to aj podľa vyššie uvedených rozhodnutí. V prvom rade však podľa rozsahu požiadaviek, ktoré treba testovať a overiť. Na tomto mieste si znovu dovolím vysloviť pochybnosť. Dokáže manažér testov odhadnúť testovacie úsilie potrebné pre jednotlivé časti systému/subsystémy? Dokáže alokovať správny počet ľudí pre jednotlivé subsystémy/časti? Nedokázal by to lepšie vedúci produkčného tímu pre danú časť/subsystém?

Zviazanosť tímov

Testovací a produkčný tím sú navzájom závislé. Ich závislosť je podľa [1] informačná a servisná. To znamená, že medzi oboma tímami dochádza k výmene informácií a služieb. Konkrétne testovací tím potrebuje od produkčného tímu tieto údaje a služby:

- Špecifikácia požiadaviek a návrh
- Použitie produktu/používateľská príručka
- Kritické požiadavky, napr. život ohrozujúce v doméne zdravotníctva
- Kritické časti návrhu, časti systému, ktoré majú väčšie riziko chyby
- Plán prác, teda už čo bolo a čo ešte bude implementované
- Požiadavky na zmeny
- Stav reportovaných chýb
- *Služby*
- Konzultovať a vytvárať návrh testov
- Zapojiť testerov do rozhodovania vecí, ktoré majú vplyv na ich prácu
- Usilovať sa o pochopenie testovacieho procesu

V opačnom garde produkčný tím potrebuje od testovacieho tímu tieto údaje a služby:

- Identifikované problémy a chyby
- Podrobný opis chyby - aká postupnosť krokov viedla k chybe
- Testovací cyklus a jeho aktuálny stav, plán testov
- *Služby*
- Podávať aktuálne informácie pre vývoj o chybách
- Usilovať sa o pochopenie produkčného/vývojového procesu
- Usilovať sa o pochopenie produktu, jeho návrhu a funkcionality

- Byť v súlade s plánom a nezdržiavať projekt

Z uvedeného zoznamu a zo skúseností z praxe by som identifikoval ich zviazanosť ako procesnú [2] a to na základe dvoch dôvodov:

- Tvorba automatizovaných testov
- Testovací cyklus

Tvorba automatizovaných testov

Tvorba automatizovaných testov je proces, na ktorom sa podieľajú obidva tímy. Pričom každý z nich vkladá do výsledku striedavo svoj podiel práce. Produkčný tím robí návrh a schvaľovanie implementácie testov, testovací tím robí implementáciu testov. Pričom návrh a schvaľovanie testov môže mať ústnu alebo písomnú formu.

Testovací cyklus

Testovací cyklus je dôležitou súčasťou procesu testovania. Realizuje sa v každej fáze testovania (viď kapitola Proces testovania). Slúži na postupné odhalenie a opravu chýb v každej fáze/úrovni testovania. Podľa [4] sa skladá z týchto krokov:

1. Spustenie testov
2. Správa o chybách
3. Revízia testov
4. Pridanie nových testov na spustenie
5. Korekcia chýb

Testovací cyklus podobne ako tvorba automatizovaných testov je proces, ktorý sa realizuje vo vzájomnej spolupráci oboch tímov. Ak sa naozaj jedná o tak vysoký stupeň zviazanosti ako predpokladám, nie je rozumnejšie dané tímy zlúčiť?

Organizácia testovacieho tímu

Kto môže realizovať testovanie? Vo všeobecnosti to môžu byť tieto štyri skupiny ľudí/role:

- Vývojári projektu (produkčný tím)
- Vnútorné zdroje projektu – testeri projektu (projektová topológia)
- Externé zdroje projektu – testeri spoločnosti (funkcionálna topológia)
- Externé zdroje firmy – dodávatelia napr. vo forme outsourcingu

V práci [3] boli analyzované a porovnávané dva spôsoby organizácie tímov v rámci projektu a ich dôsledky. Pričom uvažovali len posledné tri menované skupiny. Práca vychádza z reálneho projektu, kde boli postupne použité a porovnané obidve organizácie tímov z rôznych hľadísk.

Samostatný testovací tím

Ide o tím zložený výlučne z testerov, zvyčajne umiestnených na jednom mieste. Takéto zloženie tímu umožňuje jeho členom ľahko diskutovať o záležitostiach testovania a jeho problémoch, čím sa zvyšuje kvalita testovania a ostatných príbuzných kompetencií. Avšak okrem týchto znalostí musí tester poznať aj testovanú obchodnú doménu ako aj návrh softvéru. Tieto znalosti musí získať buď od klienta v danej obchodnej doméne alebo od vývojárov.

Dôvodom častého používania samostatného tímu je podľa [3] používanie vodopádového modelu (V-modelu) vývoja v minulosti. V ňom sa celé testovanie sústredilo na koniec vývoja a z toho dôvodu sa použil buď samostatný testovací tím alebo pôvodný tím vývojárov. Na zmiešaný tím nebol dôvod, pretože vývoj a testovanie boli oddelené.

Ďalším dôvodom používania je fakt, že samostatný tím umožňuje lepšie – nie duplicitné riadenie a správu testovacích artefaktov. To znamená, že požiadavky, plány, dokumentácia, skripty a výsledky (všetko s prívlastkom testovací) sú zdieľané na jednom mieste.

Zmiešaný tím

Zmiešaný tím je tím zložený z ľudí viacerých rolí/funkcií (cross-functional) – napr. návrhári, programátori, testeri a samotný klient. Zmiešané tímy boli v praxi použité ako "feature - oriented" tímy na implementáciu konkrétnej vlastnosti systému. Teda tento typ tímu je orientovaný skôr na používateľa/oddeliteľnú záležitosť ako na konkrétny modul systému.

Zmiešaný tím môže svojim obsadením pokryť celý životný cyklus softvéru – celý vývoj. Výhodou je tiež jeho nezávislosť od iných tímov v oblasti riadenia a vývoja. Preto sa podľa mňa tento typ tímu hodí hlavne na komponentovo – orientovaný vývoj. Teda keď systém môže byť dekomponovaný na stabilnú množinu komponentov, ktoré implementujú a testujú zmiešané tímy, jednu komponentu na jeden tím.

Výhody a nevýhody zmiešaného tímu

V tejto kapitole zhrniem dosiahnuté prínosy a nedostatky nasadenia zmiešaného tímu, ktorý nahradil samostatný tím v projekte podľa [3]. Výhody aj nevýhody sa objavili v niekoľkých oblastiach. Pre účely zabezpečenia kvality sú najdôležitejšie: proces testovania, kvalita produktu a efektivita testov, ktoré ďalej zhrniem.

Kvalita produktu a efektivita testov

- Zvýšila sa efektivita testov počas vývoja
- Zvýšila sa kvalita produktu
- Celková efektivita testov v projekte sa nezmenila
- Zhoršila sa kvalita kódu

Proces testovania

- Testeri sú menej závislí od dokumentácie

- Zlepšil sa vplyv testerov na návrh softvéru a jeho vývoj
- Zlepšila sa reakcia testerov na zmeny. Testeri reagujú na zmeny rýchlejšie, pretože "zdroj zmeny" je súčasťou ich tímu
- Predĺžil sa čas vykonávania testov skrátením prípravy testov, čo je ale na druhej strane nevýhoda. Inak povedané, nová organizácia tímu priblížila proces testovania agilným metódam so skorým štartom testov.
- Skoré testovanie má nevýhodu v odhaľovaní známych chýb príliš skoro, ešte pred ich implementáciou.

Záver

V tejto práci som sa snažil spochybniť zaužívané miesto a postavenie, ktoré má samostatný testovací tím s centrálnym riadením v projektoch veľkého rozsahu. Na niektorých miestach sa mi to viac či menej podarilo. Niektoré kapitoly boli ale iba čisto teoretické, aby vniesli trochu základov do problematiky testovania a zabezpečenia kvality ako takej. Nechávam na čitateľovi, nech si sám utvorí názor. Tejto práci ostávam dlžný ešte prezentovať výhody pre samostatný testovací tím.

Použitá literatúra

1. Bach, J.: *Managing a Software Test Team*, http://www.sqatester.com/qateam/managingtestteam.htm, navštívené 17. 10. 2008
2. Bieliková, M.: *Softvérové inžinierstvo: Princípy a manažment*. Slovenská technická univerzita, Bratislava, 2000.
3. Bruhn-Pedersen, M.: *Test Team Organisation in Functional and Cross-functional Teams*. Master thesis EBUSS, 26-34, 82-86, 124-124
4. Hopko, J.: Testovanie a kvalita v softvérovom projekte. In: *Zborník esejí FIIT STU* 2008, 6-7.
5. Patton, R..: *Software testing*. Sams Publishing, Inc. 2004.

Annotation

Is there really a separate test team needed?

This question niggled my mind because of my practice. To be specific what practice, it is my short-term occupation in a test team. You could oppose that my short-term, approximately two months long practice does not give me the right to express doubts and give judgements. However, quite the contrary may authorize me to this, if I see the imperfection so soon afterwards. I will summarize them. Test team is an odd expression for something that can not be called a team. After all, a team is mainly about co-operation, which leads to bigger utility than amount of utilities of individuals. I do not see this in the test team. Outcomes of one member of

the team are not used and if they are used, they are used only in limited way. Every member of the team works only for himself, he tests manually or he writes automatic tests for requirements assigned to him. It means that cohesion of the test team is low and that is a problem. If every tester tests his own group of requirements, would not be better to subjoin him to the developer team that implements the same requirements? Is it not better for him to be closer to the source and to the potential changes of the production code? Nevertheless, there are two sides of the same coin. I will try to summarize pros and cons of an individual test team.