KONTROLA KVALITY. NA KONCI? POČAS.

„Úspech má mnoho otcov, neúspech je sirota."
(Príslovie)

Ján Chlpek

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

Abstrakt. *Pri pojme kontrola kvality si bez hlbšieho premýšľania mnoho ľudí predstaví kontrolu kvality hotového výrobku. Softvér je tiež určitým druhom výrobku a naskytuje sa otázka: Stačí overiť a skontrolovať vlastnosti výrobku, splnenie požiadaviek klienta až pri finálnej verzii produktu? Ako už iste tušíte, takýto prístup nie je správny a preto existuje mnoho metód zabezpečenia kvality softvérového produktu počas celého vývoja, ktoré si v eseji krátko priblížime a zamyslíme sa nad nutnosťou a prínosmi niektorých týchto metód. Zameriame sa na prístupy založené na vzájomných prehliadkach (peer reviews) a podporíme názor, že zabezpečenie kvality nie je len o testovaní, ale že je to komplexná záležitosť nevyhnutná pre úspešnosť softvérových projektov.*

Kľúčové slová: *kvalita, testovanie, prehliadky, inšpekcia, peer reviews*

Zabezpečenie kvality

Nie všetkým je známe, čo sa pod týmto slovným spojením skrýva. Kvalitu softvérového projektu poznáme najmä v súvislosti so splnením požiadaviek klienta. Jednoducho, ak aj nie je produkt zmysluplný, ale zákazník si to presne takto prial, produkt teda spĺňa všetky požiadavky klienta, môžeme ho označiť za kvalitný. V tabuľke 1 uvádzame jedny z aspektov kvality definovaných iným autorom [4]. Všetky tieto vlastnosti softvéru by mali

byť na čo najvhodnejšej úrovni a čo sa týka rovnováhy, tá závisí od typu a určenia softvéru, nie na každý produkt sú rovnaké požiadavky, u niektorých môže byť absencia alebo slabšie prepracovanie určitej vlastnosti akceptovateľné.

Tab. 1. Kvalitatívne vlastnosti softvérového produktu.

Externá kvalita	**Interná kvalita**	**Budúca kvalita**
Korektnosť	Výkonnosť	Flexibilita
Spoľahlivosť	Testovateľnosť	Znovupoužiteľnosť
Použiteľnosť	Dokumentácia	Udržovateľnosť
Integrita	Štruktúra	

Do stavu, kedy môžeme produkt považovať za kvalitný, sa dostaneme cez mnoho podprocesov celého procesu vývoja, ktoré pri väčších projektoch ani zďaleka nie sú jednoduché. Vieme, že čím komplikovanejšie a zložitejšie procesy sú, čím viac ľudí je do problému zahrnutých, tým je väčšia šanca, že sa v ktorejkoľvek etape vývoja vyskytnú chyby. Zatiaľ nás nezaujíma, aké chyby to sú, aký majú pôvod. Chceme poukázať na to, že nedokonalosti vznikajú všade a nie je proces, nie je softvérový alebo iný projekt, v ktorom by sa počas vývoja nevyskytla žiadna chyba, bug, zlyhanie jednotlivca, nepozornosť alebo chyba zámerne prehliadaná. Takéto, na prvý pohľad nevinné chyby, môžu mať v konečnom dôsledku pri zavedení produktu do reálnej prevádzky fatálne následky, takže je veľmi nežiadúce, niekedy až nemysliteľné, aby akýkoľvek softvér obsahoval nezvládnuté časti bez snahy dať ich do poriadku.

Detekcia chýb objavených až na konci vývoja softvéru nie je vhodné, pretože náklady na odstránenie v takomto prípade sú niekoľko desiatok až stoviek krát vyššie ako keby bola chyba objavená skôr, v ideálnom prípade ihneď po „zhrešení“ či už programátora, analytika, návrhára alebo iného účastníka softvérového procesu. Preto existuje mnoho metód, techník a prístupov, ktoré zabezpečujú kvalitu softvéru, čiže podporujú odstránenie nedostatkov vyvíjaného produktu včas. Včas myslíme tak, že finančné náklady alebo nároky na čas a úsilie sú minimalizované, v horšom prípade sa o nich dá povedať, že sú omnoho menšie ako bez ich využitia.

Pravdepodobne najznámejšou doménou týkajúcou sa zabezpečenia kvality je testovanie. Síce poznáme mnoho druhov testovania a testov, ale aby sa projekt po stránke zabezpečenia kvality blížil k ideálnemu stavu, je potrebné použiť aj mnohé ďalšie metódy a postupy, ktoré sú často kľúčové a v rôznych etapách vývoja aj užitočnejšie ako testovanie. Týmto nechceme povedať, že testovanie je zbytočné! Testovanie, rovnako ako aj ostatné postupy, môže odhaliť problémy inými metódami neodhalené a naopak sa pri testovaní môžu prehliadnuť iné dôležité chyby, ktoré vznikli počas vývoja. Preto sa treba pozerať na zabezpečenie kvality komplexne, neobmedzovať sa na použitie iba niekoľkých technológií, ale pokúsiť sa zabezpečovať kvalitu najlepšie ako sa dá a nezanedbať túto často zanedbávanú (hlavne pri menších projektoch a v menších firmách, kde nie sú priamo na to určení ľudia) súčasť vývoja softvérového produktu.

Trochu sme si priblížili pohľad na zabezpečenie kvality a môžeme poznamenať, že to teda „nie je samostatná technológia, ale metóda a prístup.“ [2] Prakticky z toho vyplýva, že

hocijaký výstup hocijakej etapy vývoja by mal byť pravidelne kontrolovaný a monitorovaný.

Na nasledujúcich stranách si krátko priblížime doteraz len spomenuté postupy zabezpečenia kvality a budeme sa venovať ich významu pre úspešnosť a hlavne zníženie nárokov na financie a vynaložené úsilie ľudí angažovaných do vývoja softvérových projektov. Zameriame sa hlavne na zhodnotenie prístupov súvisiacich s prezeraním a inšpekciou softvéru, čo je jedna rozsiahla a dôležitá súčasť kontroly a zabezpečenia kvality softvéru.

Začnime od testovania

Testovanie sme umiestnili na začiatok a rozhodli sme sa tak preto, lebo sa mu nebudeme venovať príliš do hĺbky. Testovanie môžeme zjednodušene definovať ako overenie požadovaného a očakávaného správania systému a nemyslíme tým iba správanie týkajúce sa správneho výstupu na daný vstup a funkcionálnych požiadaviek, ale takisto môžeme testovať nefunkcionálne požiadavky systému, napríklad správanie sa pri neočakávaných situáciách, veľkom nápore používateľov, dát, nevhodne použitom hardvéri a pod.

Existujú aj postupy vývoja softvéru, ktoré sú priamo závislé od testovania, sú to tzv. „test-first" prístupy. Najznámejším je TDD (Test Driven Development), čiže vývoj riadený testovaním, kde si programátori najskôr vytvoria test a následne k nemu naprogramujú časť kódu, ktorá bude spĺňať ohraničenia dané testom. Takýto vývoj je veľmi špecifický a vyžaduje tvorbu veľkého množstva testov. V [1] je problematika prínosov a efektivity pri využívaní týchto postupov podrobne opísaná a uvádza sa tu, že pri TDD programátor napíše viac testov, ako by bolo napísaných bez použitia TDD, to znamená, že to vedie k vyššej úrovni produktivity, podporuje to lepšiu dekompozíciu a pochopenie požiadaviek a znižuje v konečnom dôsledku potreby ladenia a prerábania práce.

Automatizované testovanie tiež neobchádza záujem vývojárov softvéru a je všeobecným prianím aby celý proces testovania v budúcnosti bol automatizovaný. Predsalen je takýto spôsob testovania v konečnom dôsledku i lacnejší a rýchlejší, pretože automatický systém dokáže vykonať mnoho krát viac testov, v určitých prípadoch rádovo milióny krát viac testov, za jednotku času ako manuálne testujúci človek. Snahy vyústili do niekoľkých systémov na automatizáciu testovania, ale tie nie sú dokonalé a zahŕňajú len určité druhy testov pre určité typy systémov.

Testovanie je však vo väčšine prípadov možné až pri spustiteľných častiach zdrojových kódov a preto nie je pre efektívne zabezpečenie kvality postačujúce. Existuje teda nejaká forma zabezpečenia kvality, ktorá zahrnie odhalenie nedostatkov, na ktoré testovanie nemá dosah?

„Peer reviews" alebo vzájomné prehliadky

Vzájomné prehliadky (angl. peer reviews - PR) sú všeobecne označované ako komplement testovania v rámci zabezpečenia kvality. S týmto tvrdením si dovolíme nesúhlasiť keďže existujú ďalšie metódy na zabezpečenie kvality, ako napríklad neustála integrácia [3], ale je pravdou, že PR poskytujú pozitívny prínos širokým záberom v odhaľovaní nedostatkov, ktoré by sme testovaním ťažko hľadali.

Pri vzájomných prehliadkach zväčša ide o to, že do projektu sú privolaní ďalší ľudia, ktorí sa na samotnom vývoji priamo nepodieľajú, teda majú na vec trochu iný pohľad ako samotný programátor na svoj kód. Ale pozor, PR nie sú zamerané na overovanie a kontrolovanie len zdrojového kódu, ale prínosom je, že nimi môžeme sledovať vývoj a kvalitu všetkých súčastí projektu od špecifikácie požiadaviek, cez analýzu, návrhy, implementáciu a pod. Tieto prehliadky môžu byť vykonávané formálnym spôsobom s prísne stanovenými pravidlami, môžu byť menej formálne s rôznou organizáciou alebo silno špecifické napríklad v tom, že na vývoji jednej súčasti sa podieľajú naraz dvaja vývojári, ktorí sedia vedľa seba a navzájom vychytávajú chyby toho druhého. Poznáme teda rôzne formy PR, ktoré si v krátkosti priblížime a zamyslíme sa nad ich pozitívami a nedostatkami.

V článku [6] autor uvádza nasledovné definované typy PR:

- **Inšpekcia** (Inspection)
- **Tímové prehliadky** (Team reviews)
- **Rekapitulácia** (Walkthrough)
- **Párové programovanie** (Pair programming)
- **Dozor jedného špecialistu** (Peer deskcheck)
- **Dozor viacerých ľudí** (Passaround)

Inšpekcie sú najformálnejšou metódou v rámci PR a majú presne a prísne definovaný priebeh a organizáciu. Jedine pri inšpekciách priebeh pozostáva z niekoľkých etáp, ktoré sú nemenné a každý účastník má svoju špecifickú rolu. Preto by mali byť najspoľahlivejším, ale aj najdrahším typom zo spomínaných. Inšpekcie sa tiež riadia viacerými odporúčanými metódami, najznámejšou je Faganova metóda. Ako aj ostatné, aj táto metóda má presne definované kroky a roly účastníkov. Technika detekcie chýb pozostáva z kontroly doterajšej práce, spísania zoznamu nájdených chýb a následného prediskutovania s autorom. Dôrazom je teda skoré odhalenie chýb v už vytvorenej rozpracovanej práci.

Podobným spôsobom prebiehajú aj tímové prehliadky, avšak pri nich sa nelipne tak na formálnosti a prísnosti dodržiavania pravidiel a presne definovanom rozvrhu, takisto sa nezakladá ani na presnom rozdelení rolí podľa nejakej schémy, čiže niektoré roly môžu byť vynechané, zlúčené a podobne, záleží na veľkosti a organizácii tímu. Takýto spôsob zabezpečovania kvality bol vhodný využiť aj v tímových projektov na našej fakulte, je však otázne, koľko času a úsilia sú študenti schopní a ochotní investovať do zabezpečovania kvality projektu a či to nebude pri nedostatku času a pracovných síl na úkor produktivity samotného projektu.

Rekapitulácie sa odlišujú od predchádzajúcich typov v tom, že hlavné slovo pri stretnutiach má autor, ktorý predstavuje svoju doterajšiu vykonanú prácu, poukazuje na problémy, ktoré sám odhalil a účastníci sa snažia takýmto spôsobom odhaliť nedostatky a diskutujú o nezrovnalostiach. Keďže táto metóda je oveľa menej formálna a účastníci sa nedostávajú do veľkej hĺbky problematiky, je otázne, na koľko sú rekapitulácie pri zabezpečení kvality efektívne a či nevyústia len do akejsi prezentácie autora bez toho, aby sa účastníci dostali hlbšie do obrazu a detailnejšie sa zoznámili s produktom.

V extrémnom programovaní (XP) sa stretávame s programovaním v pároch. Párové programovanie môže tiež nie zanedbateľne prispieť k zabezpečeniu kvality, kedy dvaja programátori pracujú spolu a navzájom vychytávajú chyby a nedostatky toho druhého. Ako sa hovorí, viac hláv, viac rozumu. Veľkým problémom pri párovom programovaní je absencia nezainteresovaného človeka, ktorý je schopný odhaliť chyby, ktoré autori zdrojových kódov z rôznych príčin nevidia. Takýto človek sa pozerá na problém s nadhľadom a z vlastnej skúsenosti môžeme potvrdiť, že často sa stane, že sa krvopotne snažíme nájsť chybu, trápime sa nad určitým problémom, a príde niekto iný, pozrie sa na kód a v priebehu pár sekúnd ukáže na riadok, nesprávne priradenú premennú, chýbajúce opustenie cyklu a riešenie je na svete. A toto sa môže stať nielen samostatnému programátorovi, ale i páru programátorov, ktorí majú na kód spoločný pohľad a hľadajú chybu rovnakým spôsobom.

Pri inšpekciách sa často stáva, že jeden člen inšpekčného tímu odhalí oveľa viac nedostatkov ako ostatní. Takýto človek, ktorý má tento dar a vynikajúcu schopnosť odhaliť chyby skôr ako ostatní a odhaľuje chyby, ktoré ostatní prehliadajú, je najvhodnejším kandidátom pre to, aby bol umiestnený na pozíciu, v ktorej bude robiť ako jediný dozor nad autormi pri tzv. „peer deskcheck“. Takýto neformálny typ PR pozostáva z toho, že jeden človek kontroluje a overuje prácu buď jedného, alebo pokým sa to dá zvládať, viacerých vývojárov. Nielenže ich upozorňuje na chyby, ale pri svojej práci môže na základe skúsenosti zistiť najčastejšie chyby alebo typy chýb, ktorých sa vývojári dopúšťajú, ich príčiny a následne im ich prezentovať, aby sa ich v budúcnosti vyvarovali. Takýto spôsob je, pochopiteľne, pre firmu oveľa menej finančne náročný, ako platenie celého inšpekčného tímu.

Podobným spôsobom sa môžu do kontrolného procesu zapojiť viacerí ľudia, ktorí budú kontrolovať a sledovať prácu jednotlivca. Keďže v tomto prípade viacerí ľudia sledujú a napomínajú vývojára, ten sa s postupom času môže začať spoliehať na to, že problémy odhalia oni a týmto klesne kvalita jeho práce a celkovo jeho produktivita. Zavedenie tohto spôsobu zabezpečovania kvality teda stojí za zamyslenie sa.

Prínosy PR?

Prínosy pre kvalitu produktu nechajme bokom a zamerajme sa na ostatné pozitíva zavedenia prehliadok do procesu vývoja softvéru. Zavedenie každej novej metódy alebo zmena postupu práce musí mať okrem hlavného cieľa aj vedľajšie účinky na celý proces, ktoré síce nie sú tak viditeľné a nie vždy si ich uvedomujeme, ale je zaujímavé ich sledovať. Niektoré z nich sme už načrtli, skúsme si ich priblížiť.

Pri inšpekciách sa stretnutí zúčastňuje viacero osôb, ktoré určite nie sú na rovnakej úrovni so znalosťami a zručnosťami. Rozoberaním problémov rôzneho charakteru sa každý z nich oboznámi so záležitosťami, s ktorými sa doteraz nestretol, alebo im neprikladal dôležitosť. Vidia, ako ostatní riešia dané problémy a pri tom sa učia. Táto skutočnosť je veľmi prínosná nielen pre samotného jednotlivca, ale aj pre celý projekt, pretože v budúcnosti už jednotlivci budú skúsenejší a schopní pozerať sa na problém z rôznych hľadísk. Nejedná sa len o detaily a štýly v rámci programovania a programovacích jazykov, ale aj o architektonické pohľady a rôzne pohľady na návrh, štýly

dokumentovania práce, poznatkov a hlavne prehlbovanie si znalostí z oblasti, pre ktorú sa daný softvér vyvíja.

Všetky typy PR poskytujú neoceniteľný prínos nie len kvalite finálneho produktu, ale hlavne celému procesu, pretože poskytujú vzájomné vedomostné obohacovanie sa členov tímov, podporujú komunikáciu a celý proces upravujú tak, aby jeho súčasti boli zrozumiteľné nielen všetkým účastníkom, ale v konečnom dôsledku je väčšia šanca, že v priebehu vývoja bude možné jednoduchšie prezentovať priebežne výsledky aj zákazníkovi.

Teraz sa venujme hlavnému prínosu PR. Pozrime sa na problém porovnaním PR a testovania. Obe metódy sú pri zabezpečení kvality produktu všeobecne uznávané a obe odhaľujú iné nedostatky. Prínos PR teda spočíva v tom, že odhalí také nedostatky v procese vývoja, ktoré nie je možné zistiť testovaním. Ak sa pýtate aké, pravdepodobne ste pozorne nečítali, alebo ste si nedávali dokopy doteraz prečítané fakty a myšlienky súvislosti aj z praktického hľadiska.

Testovanie poskytuje kontrolu kvality vykonateľných, hotových súčastí produktu. Pre testovanie je špecifické to, že overuje správanie produktu v danej situácii, ktorá je reálna a nemusí to byť práve očakávané správanie, ktoré pri PR kontrolóri vydedukujú z kódu. Môže sa však stať, že pri testovaní systému po vykonaní úkonu nastane neidentifikovaná chyba, následne sa bude hľadať v množstve kódu prislúchajúcej funkcionality a nakoniec sa zistí, že bolo len zle implementované spúšťanie funkcionality a nie samotná funkcionalita ako následok spustenia. Pri použití PR je pravdepodobné, že podobná záležitosť by bola pri prehliadke kódu nájdená.

Pomocou PR odhalíme nedostatky týkajúce sa efektivity kódu, jasnosti implementovaného problému, môžeme odhaliť nezmyselné komentáre, nadbytočné alebo opakujúce sa časti kódu a pod., neúplnú špecifikáciu požiadaviek, chyby v návrhu, môžeme zistiť, že pri súčasnom stave systému bude komplikovaná údržba atď.

V tabuľke 2 uvádzame porovnanie odhalenia chýb inšpekciami a testovaním [5].

Tab. 2. Porovnanie odhalenia chýb inšpekciou a testovaním.

Typ chyby	Inšpekcia	Testovanie
Chyby rozhraní modulov	Áno	Nie
Nadmerná zložitosť kódu	Áno	Nie
Prítomnosť nevyžiadanej funkcionality	Áno	Nie
Problémy s používaním	Nie	Áno
Problémy s výkonom	Áno	Áno
Zle štruktúrovaný kód	Áno	Nie
Nesplnenie požiadaviek	Áno	Áno
Chyby ohraničení hodnôt	Áno	Áno

Testovanie a PR sa teda navzájom dopĺňajú a existujú i ďalšie formy zabezpečenia kvality, akým je napríklad neustála integrácia hotových častí. Táto sa využíva hlavne pri veľkých projektoch, kedy sa na súčastiach pracuje samostatne, a vytvorené časti alebo moduly sa v určitých časových intervaloch pravidelne integrujú, aby sa včas odhalili problémy s nekonzistenciou celého produktu a problematickou spoluprácou jednotlivých častí.

Na čo si dať pozor...

Je zrejmé, že PR majú veľký prínos pri zabezpečení kvality, ale v každej sfére živného cyklu môžu nastať komplikácie. Pri používaní PR v spoločnosti je potrebné dbať na to, aby sme sa vyvarovali bežným chybám, ktoré môžu spôsobiť, že PR nám pri vývoji spôsobia viac škody ako osohu.

Často sa stáva (a nielen pri softvérových projektoch), že človek, ktorý má na starosti kontrolu kritizuje skôr vývojára ako produkt. Snaží sa ukázať, o koľko je chytrejší, ako by to on spravil lepšie. Je jasné a logické, že človek na kontrolnej pozícii je spravidla schopnejší a má viac skúseností. A preto by nemal na tento fakt poukazovať, ale mal by sa zameriavať hlavne na produkt a jeho vylepšenia, prípadne nejakým ľudským spôsobom navádzať programátora na lepšie výsledky, aby sa v nestalo, že pri plánovaných kontrolách programátor vzhľadom na obavy neukáže svoju kompletnú prácu, ale len časti, s ktorými si je istý, že nevyvolajú vlnu kritiky. Je dôležité, aby sa pri inšpekciách kontrolovali aj nedokončené súčasti, pretože čím skôr sa chyby odhalia, tým lepšie. Ak nájdeme chyby v práci, ktorá je hotová, dajme tomu, na 15%, je zrejmé, že ostatných 85% sa bude vyvíjať bez nich a kvalitnejšie.

Podobným problémom môže byť, že niektorí inšpektori často lipnú na štýle programovania, rôznych nepodstatných detailoch a podobne a ich správa o kontrole obsahuje len takéto nedostatky produktu. Ak sa pri kontrole nezaoberá podstatnými záležitosťami, ale len tým, ako kód vyzerá a unikajú mu logické, funkčné a iné chyby, je na mieste, aby ľudia na vyššej pozícii skontrolovali prácu daného človeka.

Nie len ego na strane kontroly môže uškodiť. Priveľmi a nie odôvodnene sebavedomý programátor môže tiež znižovať efektivitu vývoja, keď nebude považovať inšpekcie za dôležité, bude odmietať poskytovať výsledky svojej práce na kontrolu a bude tvrdiť, že jeho práca je dokonalá a žiadnu kontrolu nepotrebuje. Pri PR takíto jedinci môžu zaujať tvrdohlavo obrannú polohu bez snahy pochopiť návrhy a kritiku ostatných, alebo jednoducho nebude brať výsledky inšpekcie vážne a bude pokračovať v práci bez napravenia chýb. V takomto prípade je tiež potrebný zásah zo strany manažmentu. Tiež aj skupina ľudí vykonávajúca prehliadky musí mať určitú autoritu, aby mohla vyžadovať prerobenie kontrolovanej práce.

Problémov so zavedením PR do procesu vývoja je softvéru je určite oveľa viac a všetky sú založené na skúsenosti a na reálnych situáciách, preto by najlepšie a najrozsiahlejšie poznatky mohli poskytnúť tí najpovolanejší, a to ľudia, ktorí sa zúčastňujú a podieľajú na procesoch zabezpečovania kvality pomocou PR v skutočných spoločnostiach.

A čo povedať na koniec?

Na predchádzajúcich stranách sme uviedli prístupy v zabezpečovaní kvality, zamysleli sme sa a vyzdvihli sme dôležitosť integrovania PR do procesu vývoja softvéru komentovaním typov prehliadok a aj porovnaním PR a testovania, ktoré nie je jedinou cestou zabezpečenia kvality. Dospeli sme k záveru, že prehliadky počas vývoja produktu, či už inšpekcie alebo menej formálne postupy, efektívne napomáhajú úspešnosti

zaručovania kvality nielen produktu, ale aj celého projektu a jeho jednotlivých súčastí. Ako pri každom procese práce viacerých ľudí sa aj pri zabezpečovaní kvality stretávame s problémami, pričom zopár problémov aj v spojitosti s využívaním PR sme uviedli. Jednotlivým typom v rámci PR by sa dalo venovať aj hlbšie, ale naším cieľom bolo ich zhodnotiť ako celok.

Celkovo môžeme skonštatovať, že pri zabezpečovaní kvality je potrebné zaviesť do organizácie vývoja produktu čo najviac disciplíny, plánovania a neustáleho kontrolovania priebežnej práce, lebo to je kľúč k tomu, aby sme sa vyhli odhaleniu nedostatkov až vo finálnej fáze, v najhoršom prípade zákazníkom a používateľmi, čo by pre spoločnosť znamenalo veľké straty rôzneho charakteru.

Použitá literatúra

1. Erdogmus, H., Moriso, M., Torchiano, M.: *On the effectivness of the test-first approach to programming.* In: Proceedings of the IEEE Transactions on Software Engineering, January 2005.
2. Feldman, S.: *Quality Assurance: Much More than Testing.* In: ACM QUEUE, January 2005.
3. Fowler M., Foemmel, M.: *Continuous integration.* September 2005.
4. Hetzel, W. C.: *The Complete Guide to Software Testing*. 2nd ed. In: Wellesley, M. : QED Information Sciences, 1988. ISBN: 0894352423. 280 s.
5. Wiegers, K. E.: *Improving Quality Through Software Inspections.* In: Software Development, vol. 3, no. 4, April 1995.
6. Wiegers, K. E.: *Seven Truths About Peer Reviews.* In: Cutter IT Journal, vol. 15, no. 7, July 2002.

Annotation

Quality assurance. At the end? Throughout.

When you hear the term quality control, without a deeper reflection, many people will understand the quality control of finished product. Software is also a kind of product and the question is: Is it enough just to verify and check the properties of the product to meet client requirements in the final version of the product? As you surely guess, this approach is not correct and there are many methods of quality assurance during the software product development, which we will aim at and we will talk about the necessity and benefits of some of these methods. We will focus on approaches based on peer reviews and support the view that quality assurance is not only about testing, but it is a complex issue essential to the success of software projects.