# AKO KVALITNE VYRIEŠIŤ TÍMOVÝ PROJEKT?

*Kvalita nie je kvantita.*

*Peter Abelovský*

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

***Abstrakt.*** *Kvalita softvéru je v súčasnosti dobre definovaný a často používaný pojem. Čo však značí pojem kvalita v ponímaní predmetu Tvorba softvérového systému v tíme, ktorý prebieha na našej fakulte? V eseji sa zaoberám výberom najvhodnejšieho pohľadu na kvalitu v kontexte tímových projektov. Vplyv na ňu majú všetky fázy životného cyklu projektu - od špecifikácie po jeho odovzdanie. Pravdepodobne najdôležitejšou pre kvalitu výsledného produktu je fáza návrhu určujúca časovú náročnosť nasledujúcich fáz projektu. Ďalším dôležitým faktorom je vzdelanie členov tímu a ich nasadenie v procese zabezpečovania kvality prebiehajúceho počas celej doby trvania projektu. Častými dôvodmi zníženej kvality nielen tímových projektov je neskoré zapojenie testovania, nesprávne použitie refaktoringu, či prílišné spoliehanie sa na návrhové vzory a moderné nástroje tvorby softvéru. V eseji vyjadrujem svoj názor k správnemu použitiu spomenutých techník v prostredí tímového projektu, pri ktorom zvyšujú kvalitu jeho výsledku.*

***Kľúčové slová:*** *kvalita softvéru, tímový projekt, testovanie, refaktoring, návrhový vzor, IDE*

## Úvod

Hoci je kvalita softvéru dobre definovaný pojem, možno sa na ňu pozerať z niekoľkých rôznych pohľadov, závisí od mnohých faktorov a je možné ju pozitívne ovplyvniť viacerými technikami a postupmi. Ako sa však pozerať na kvalitu produktu, ktorý je

výsledkom predmetu Tvorba softvérového systému v tíme? Ako jeho kvalitu ovplyvňuje kvalita celého projektu?

Tímový projekt je v porovnaní s projektmi z praxe odlišný z viacerých aspektov. Tím študentov je na rozdiel od reálnych tímov vo väčšine prípadov menší. Podobne sú menšie aj skúsenosti jeho členov s tímovou spoluprácou. Tím nemusí byť vyvážený z hľadiska schopností a skúseností svojich členov. Kvalita výsledného produktu teda priamo závisí od zloženia tímu, snahy a nasadenia jeho členov v jednotlivých fázach životného cyklu projektu.

Ďalším špecifikom tímového projektu je nemožnosť predĺžiť dobu jeho trvania. V reálnych projektoch je často aspoň čiastočne možné oddialiť odovzdanie projektu svojmu zadávateľovi.

Poslednou odlišnosťou sú rozdielne ciele tímového projektu v porovnaní s reálnymi projektmi, či už väčšími alebo tými veľkosťou podobnými tímovému projektu. Kým v reálnych projektoch je dôraz kladený na produkt, v prostredí tímového projektu je dôležitý skôr samotný proces jeho tvorby.

Jednotlivé fázy projektu nemajú na výslednú kvalitu jeho produktu rovnaký vplyv. Na ktorú časť životného cyklu by teda členovia tímu mali sústrediť nadštandardné úsilie a tým ušetriť čas a zvýšiť hodnotu výsledku? Aj na túto otázku vyjadrujem v eseji pre kontext tímového projektu svoj názor.

Dôležitým prostriedkom zvyšovania kvality každého softvérového projektu je často podceňovaný proces testovania, či už testovania v priebehu vývoja resp. toho záverečného. Ako však proces následného odstraňovania identifikovaných chýb zefektívniť? Metódami refaktoringu, teda znižovaním celkovej zložitosti, či už na úrovni návrhu resp. implementácie. S metódou refaktoringu ďalej súvisí odstraňovanie nevhodných častí kódu náchylných k chybám pomocou návrhových a implementačných vzorov. V eseji vyjadrujem svoj názor k správnemu použitiu daných techník v prostredí tímového projektu, na ktoré je vďaka všetkým jeho špecifikám potrebné klásť dôraz.

## Kvalita softvéru všeobecne

Kvalita je podľa normy ISO 8402 všeobecne definovaná ako súhrn vlastností a charakteristík výrobku alebo služby, ktoré preukazujú jeho schopnosť splniť určené alebo odvodené potreby. Norma ISO 9126 pojednávajúca o kvalite softvéru ďalej určuje primárne a sekundárne charakteristiky, od ktorých jeho kvalita závisí. Tabuľka 1. zobrazuje každú primárnu charakteristiku spolu s príslušnými sekundárnymi charakteristikami.

**Tab. 1.** Primárne a sekundárne charakteristiky kvality softvéru podľa normy ISO 9126 [5]

| Primárna charakteristika | Subcharakteristiky |
|---|---|
| Funkcionalita | Vhodnosť, presnosť, interoperabilita, bezpečnosť, vyhovenie funkcionálnym požiadavkám |
| Spoľahlivosť | Zrelosť, tolerancia k chybám, obnoviteľnosť, vyhovenie požiadavkám na spoľahlivosť |
| Použiteľnosť | Pochopiteľnosť, naučiteľnosť, pohotovosť, atraktívnosť, vyhovenie požiadavkám používateľa |

| | |
|---|---|
| Efektívnosť | Správanie sa v čase, využitie zdrojov, vyhovenie požiadavkám efektívnosti |
| Udržiavateľnosť | Analyzovateľnosť, možnosť zmeny, stabilita, testovateľnosť, vyhovenie požiadavkám udržiavateľnosti |
| Prenositeľnosť | Adaptabilita, zložitosť inštalácie, nahraditeľnosť, schopnosť koexistencie, vyhovenie požiadavkám prenositeľnosti |

Štandard ISO 9126 teda hodnotenie kvality softvéru rozdeľuje na množinu primárnych a sekundárnych charakteristík. Dané rozdelenie charakteristík však nie je podľa [5] úplne vhodné a napr. bezpečnosť by mala podľa autorov článku tvoriť samostatnú primárnu charakteristiku.

Na hodnotenie kvality softvéru sa používajú tzv. metriky, ktoré poskytujú spôsob ako získať cenné informácie o štruktúre a kvalite systému [7]. Kvantitatívne vyjadrenie kvality, ktoré metriky poskytujú, môže hodnotený softvér posudzovať z viacerých možných hľadísk. Medzi tie najpoužívanejší patria metriky hodnotiace kvalitu z pohľadu kódu (napr. počet parametrov pre modul či percento riadkov tvoriacich komentáre) a metriky hodnotiace kvalitu z pohľadu návrhu (napr. počet rozhodovacích bodov či počet volaní funkcií v rámci modulu) [4]. Je preukázané, že softvérové metriky odrážajú kvalitu softvéru a sú teda často používané v metódach pre určovanie jeho kvality [7].

## Správny pohľad na kvalitu

Aj keď je pojem kvality softvéru vymedzený pomerne presne, možno sa naň podľa [9] pozerať z viacerých možných uhlov pohľadu:

1. Z hľadiska *používateľa* dokáže kvalitný softvér dlhodobo a v maximálnej možnej miere uspokojiť jeho požiadavky.
2. Z hľadiska *produktu* je jeho kvalita závislá na súhrne vnútorných charakteristík produktu a môže byť určená na základe prítomnosti resp. absencie daných charakteristík.
3. Z hľadiska *spracovania* sa kvalitný softvér drží svojej špecifikácie a kladených požiadaviek a jeho kvalita je funkciou kvality procesu jeho tvorby.
4. Z hľadiska *hodnoty* ide o splnenie potrebnej funkcionality pri dodržaní rozumnej ceny produktu.

Keďže výsledok tímového projektu často nemá svojich stálych používateľov, je v porovnaní s projektmi z praxe ochudobnený o hodnotenie z pohľadu ich dlhodobého uspokojenia. Samozrejme, je potrebné snažiť sa o produkt, ktorý by bol v prípade nasadenia do praxe pre svojich používateľov dlhodobo prijateľný.

Výsledná aplikácia tímového projektu nebude po ukončení svojho vývoja predaná zadávateľovi. Jej hodnotenie z pohľadu pomeru funkcionality k jej cene je teda bezpredmetné. Cenu v prípade pohľadu č. 4. však možno nahradiť časom. Jedným zo správnych pohľadov na kvalitu výsledného produktu je teda hodnotenie vytvorenej funkcionality pri dodržaní zadaného termínu.

Iným vhodným spôsobom hodnotenia kvality výsledku tímového projektu je určenie súhrnu jeho vnútorných charakteristík podľa bodu č. 2. Týmito charakteristikami môžu byť práve primárne a sekundárne charakteristiky definované normou ISO 9126.

Podľa môjho názoru najvhodnejší pohľad na kvalitu v kontexte tímových projektov predstavuje pohľad č. 3. Kvalitný softvérový produkt sa drží svojej špecifikácie a kladených požiadaviek. Jeho kvalita je teda funkciou kvality procesu jeho tvorby. Proces tvorby produktu predstavuje priebeh celého tímového projektu. Kvalita tímového projektu ako takého popri iných faktoroch výrazne závisí od plnenia cieľov predmetu, v rámci ktorého projekt prebieha – častej a vecnej komunikácie v tíme či vhodného rozdelenia úloh podľa individuálnych schopností členov tímu.

## Životný cyklus projektu

V prostredí tímového projektu na našej fakulte sú agilné techniky vývoja softvéru stále v menšine. Vypracovanie projektu teda obvykle prebieha pomocou klasických modelov životného cyklu softvéru. Vplyv na jeho kvalitu majú všetky fázy cyklu, no pravdepodobne najvýraznejšie sa prejavuje vplyv fázy návrhu. Fáza návrhu vychádza zo špecifikácie, v ktorej je potrebné na vhodnej úrovni abstrakcie určiť požiadavky, od ktorých sa odvíja samotný návrh.

Dobrý návrh dokáže zredukovať čas a úsilie potrebné na vytvorenie riešenia. Pri reálnych projektoch z praxe navyše aj finančné náklady. Pri výbere technológií, je potrebné dbať na schopnosť splniť pomocou nich všetky špecifikované funkcionálne aj nefunkcionálne požiadavky. Zároveň treba brať ohľad na čas a úsilie, ktoré je potrebné pri ich použití vynaložiť na implementáciu potrebnej funkcionality. Získanie času navyše v pevne ohraničenej dobe trvania tímového projektu je pre proces zabezpečovania kvality veľmi výhodné.

Téma, ktorú má pridelenú môj tím bola v predmete tímového projektu riešená aj v minulom roku. Pre implementáciu riešenia však boli zvolené v podstate nekompatibilné technológie. Členovia tímu tak museli riešiť problémy so zložitosťou komunikácie jednotlivých modulov aplikácie. Čas im zaberalo vytváranie prepojení medzi modulmi, ktoré kvalitu aplikácie nezlepšovali. Ak by však zvolili kompatibilnejšie technológie, ušetrený čas mohol byť použitý na testovanie prípadne doplnenie novej funkcionality.

Návrh architektúry systému by mal byť podľa môjho názoru čo možno najjednoduchší, zároveň však každé zjednodušujúce rozhodnutie by malo byť dobre zdôvodniteľné. Ak návrh prebieha prostredníctvom objektovo-orientovaného prístupu, je potrebné na kvalitu podrobného návrhu klásť väčší dôraz v porovnaní s implementáciou projektu pomocou procedurálnych jazykov. Keďže pri použití objektovo-orientovaného návrhu je potrebné počítať s možnosťou použitia návrhových a implementačných vzorov.

## Projekt tvoria ľudia

Kvalita softvérového projektu vrátane toho tímového závisí vo veľkej miere od schopností a prístupu jednotlivých členov tímu. Schopnosti členov tímu predstavuje na jednej strane množstvo ich k téme relevantných vedomostí, na strane druhej ich predchádzajúce skúsenosti s používanými technológiami a nástrojmi. V konečnom dôsledku sa v zložitosti

návrhu riešenia prejavujú analytické schopnosti tímu, v čitateľnosti a možnosti úpravy kódu zas programátorské schopnosti a skúsenosti.

Výskum v [1] zobrazuje koreláciu medzi schopnosťami tímu a kvalitou produktu v jednotlivých fázach životného cyklu. Výskum bol realizovaný pre menšie projekty používajúce objektovo-orientovanú paradigmu, teda projekty veľkosťou aj použitím objektovo-orientovanej paradigmy podobné mnohým tímovým projektom. Najvýraznejšie sa korelácia medzi schopnosťami tímu a kvalitou produktu prejavuje vo fáze implementácie. Teda zapojenie menej skúsených programátorov má výrazný negatívny účinok na kvalitu produktu, kým pri zapojení tých skúsených sa jeho kvalita zvyšovala.

Pri výbere technológií je podľa môjho názoru potrebné vhodne zhodnotiť predchádzajúce skúsenosti členov tímu s vybranými technológiami. Úlohy v tíme je následne nutné rozdeliť podľa zistených skúseností, teda tým menej skúseným zveriť tie jednoduchšie. Najnebezpečnejšie je priradiť náročnejšie úlohy programátorom nachádzajúcim sa niekde medzi základným a mierne pokročilými skúsenosťami s danou technológiou. V tomto prípade je zároveň najnáročnejšie odhadnúť mieru ich skutočných skúseností. Títo členovia tímu môžu chybami v procese implementácie výrazne znížiť kvalitu produktu, keďže im budú pravdepodobne zverené aj náročnejšie úlohy.

## Splnené ciele predmetu verzus kvalitný projekt

Ako som spomenul vyššie v texte eseje, na kvalitu výsledku tímového projektu sa možno pozerať ako na kvalitu procesu jeho tvorby. Proces tvorby softvérového systému predstavuje samotný tímový projekt. Na základe akých kritérií však zhodnotiť kvalitu samotného projektu? Jedným z možných uhlov pohľadu je zhodnotenie splnenia cieľov predmetu – množstva získaných skúseností s prácou v tíme, preukázania schopnosti dorozumieť sa, rozdeliť si úlohy, teda pripraviť sa na prácu na projektoch väčšieho rozsahu. Vytvorenie samotného produktu je v prípade tímového projektu v rebríčku dôležitosti cieľov nižšie.

Splnenie cieľov predmetu je teda s vytvorením kvalitného produktu vzájomne previazané. Kvalitný produkt ťažko vytvoríme bez častej, vecnej komunikácie a dobrého rozdelenia úloh na základe individuálnych schopností členov tímu. Na druhej strane vznik dobrého produktu implikuje pravdepodobne správne fungovanie tímu. Dôraz na proces tvorby potvrdzujú aj kritéria hodnotenie predmetu, kedy hodnotenie produktu a jeho funkcionality tvorí len časť z celkového hodnotenia. Druhú časť tvorí hodnotenie práce v tíme, riadenia tímu a jeho zdokumentovanie.

Správne fungovanie tímu možno pozorovať aj na množstve a kvalite konzultácií k jednotlivým alternatívam riešenia problémov v rámci projektu. Konzultácie v tíme by sa mali týkať aj už vykonanej práce, kedy nový pohľad na vec môže odhaliť návrhové a implementačné chyby. Konzultácie vykonanej práce taktiež umožňujú lepšie spoznanie celého projektu jednotlivým členom tímu vrátane častí, na ktorých priamo nepracujú. Na slovo „verzus“ v nadpise tejto časti sa teda možno pozerať aj ako na znak ekvivalencie, ktorý podľa môjho názoru správne vystihuje vzťah medzi cieľmi predmetu a kvalitou projektu.

## Testovanie nikoho nebaví

Testovanie je pokladané za základný prostriedok zvyšovania kvality nielen softvérového produktu. Cieľom testovania je overenie správania sa aplikácie či jej modulu s ohľadom na splnenie špecifikácie a používateľských požiadaviek. Jeho ďalšou úlohou je odhalenie prípadnej chybnej funkcionality. Testovanie zastrešuje viacero rozličných aktivít, od testovania samostatných jednotiek aplikácie, ktoré vykonáva programátor v priebehu implementácie až po akceptačné testovanie u zákazníka [2].

Proces testovania je teda základnou aktivitou pre zabezpečenie splnenia primárnych charakteristík funkcionality a spoľahlivosti podľa normy ISO 9126. Toto platí napriek všetkým jeho špecifikám oproti reálnym projektom aj pre tímový projekt bežiaci na našej fakulte. Prečo je však proces testovania často podceňovaný? Možno nie je ani tak podceňovaný ako zanedbávaný. Študenti (vrátane mňa) sa mu obvykle vyhýbajú. Veď pridávanie novej funkcionality do aplikácie je omnoho zaujímavejšie! Často sa mi stáva, že po pridaní novej funkcionality odskúšam jej fungovanie len pre základný scenár jej použitia a už premýšľam nad pridávaním ďalšej. Menej je niekedy viac. Správanie sa nového modulu či funkcie je potrebné overiť aj vzhľadom na jej prepojenia s už existujúcimi časťami aplikácie. Často však nemožno týmto spôsobom odhaliť všetky nedostatky, pretože testovaná časť nie je zaradená do svojho konečného kontextu. Po vytvorení konečného kontextu danej funkcie či modulu je teda nutné opäť otestovať korektnosť jej správania sa.

Ďalším špecifikom tímového projektu je fakt, že návrh a testovanie môže vykonávať ten istý člen tímu. Ak budú určité súvislosti prehliadnuté pri návrhu, je veľmi pravdepodobné, že budú prehliadnuté aj v procese testovania a tak môžu spôsobiť chybné správanie sa aplikácie [3]. V projektoch, na ktorých som pracoval samostatne, som tú istú chybu často prehliadol aj niekoľkokrát. Odhaliť sa mi ju podarilo až keď som sa k danej časti kódu alebo návrhu vrátil s určitým odstupom času, resp. som bol nútený posúdiť ho z iného uhlu pohľadu. Tu sa opäť prejavuje potreba konzultácií vykonanej práce s ostatnými členmi tímu, ktorí by k nej mali povedať svoje pripomienky a názory. Vhodné je nielen konzultovať, ale vykonanú prácu, či už zdrojový kód alebo diagramy predviesť na stretnutí. Autor by mal obhájiť použité riešenia a dôvody vyradenia možných alternatív.

Testovanie je teda nutné tak pre produkt tímového projektu, ako i pre každý iný softvérový produkt. Na tento proces je však potrebné vyhradiť dostatočné množstvo času. Tu sa prejavuje ďalšia výnimočnosť tímového projektu a tou je pevný termín jeho ukončenia. Jeho trvanie nemožno v žiadnom prípade predĺžiť. Veľmi vhodné je podľa môjho názoru testovať vytváranú aplikáciu priebežne. Čas potrebný na jej konečné testovanie je náročné odhadnúť alebo naplánovať. Iným dôvodom pre skoré začatie testovania je fakt, že náprava chýb v skorších štádiách životného cyklu stojí nepomerne menej nákladov. V prostredí tímového projektu sú nákladmi čas a úsilie členov tímu. Aj pri rozhodovaní sa medzi testovaním a pridávaním novej funkcionality teda platí – radšej kvalita ako kvantita.

## Je to skutočne refaktoring?

Ďalšou z techník umožňujúcich zlepšovanie návrhu a implementácie softvérových systémov je refaktoring. Je definovaný ako proces úpravy softvérového produktu spôsobom, ktorý neovplyvňuje jeho externé správanie sa, ale len zlepšuje jeho vnútornú štruktúru. Cieľom je znížiť komplexnosť tak na úrovni návrhu ako aj zdrojového kódu. To umožňuje znížiť náklady a zmenšiť priestor pre návrhové chyby [7]. Zníženie zložitosti zahŕňa aj zníženie duplicity v zdrojových kódoch a návrhu. Táto najčastejšie vzniká pri kopírovaní určitej časti a následnej miernej modifikácií.

Refaktoring teda vychádzajúc z jeho definície zlepšuje najmä udržiavateľnosť aplikácie. Opis jeho cieľov ho priamo definuje ako prostriedok pre zvyšovanie kvality softvérového produktu. Aké sú však skutočné dôsledky jeho použitia? A aký vplyv má na zostávajúce charakteristiky kvality podľa normy ISO 9126? Podľa [7] je vplyv refaktoringu na vybrané metriky určujúce kvalitu softvérových projektov nepriaznivý. Experiment sa týkal tak procedurálnych aplikácií, ako aj aplikácií vytvorených pomocou objektovo-orientovaného prístupu. Autori článku však dôvody nepriaznivého účinku refaktoringu na kvalitu neuvádzajú. Podľa môjho názoru sa pri revíziách softvérových produktov často pridávajú nové prvky namiesto ich uberania, pridávajú sa nové prepojenia medzi modulmi a triedami namiesto ich redukcie. Vykonávané zmeny teda často nie je vhodné nazvať refaktoringom.

S podobnými problémami som sa stretol aj ja. Mnohokrát bol môj kód po nových zásahoch dlhší, zložitejší a viac previazaný. Uvedené problémy môžu vzniknúť najmä pri používaní techniky refaktoringu menej skúsenými programátormi. To je obvykle prípad aj tímového projektu. Aplikácia ktorá je jeho výsledkom sa tak môže po zmenách, ktoré ich autori možno nazvú refaktoringom, stať viac previazanou a menej súdržnou, keďže jednotlivým modulom bude priradenej viac funkcionality. Aké z toho plynie ponaučenie? Pri zámere použiť refaktoring je potrebné sledovať jeho základný cieľ – zníženie komplexnosti na úrovni návrhu a implementácie.

## Vyriešim to návrhovým vzorom

Návrhové resp. implementačné vzory často na fázu refaktoringu priamo nadväzujú. Poskytujú riešenie pre často sa opakujúce problémy v prostredí návrhu a implementácie softvéru. Ich cieľom je popri zvýšení kvality návrhu a teda celého systému aj zjednodušenie jeho budúceho rozširovania [6]. Napriek ich cieľu však autori v [6] poukazujú na nepriaznivý vplyv použitia skúmaných návrhových vzorov na metriky hodnotiace kvalitu softvéru, napriek faktu že dokázali vyriešiť návrhové problémy. Autori uvádzajú aj špecifické dôvody zníženia kvality pre každý z posudzovaných vzorov.

Všeobecným problémom návrhových vzorov je fakt, že napriek ich schopnosti zjednodušiť a zrýchliť proces návrhu, nepriaznivo ovplyvňujú určité primárne charakteristiky kvality. Ide najmä o subcharakteristiky použiteľnosti a to pochopiteľnosť a naučiteľnosť. Autor teda môže použitím návrhového vzoru sťažiť prácu inému členovi tímu, ktorý bude v jeho práci pokračovať. V prostredí tímového projektu je málo pravdepodobné, že každý člen tímu má dostatočné vedomosti z oblasti návrhových vzorov. Preto je potrebné poznať ich negatívne účinky a brať na ne pri ich používaní

ohľad. Čas získaný zavedením návrhového vzoru môže byť teda ľahko stratený pri snahe pochopiť danú časť systému iným členom tímu. Opäť, podobne ako pri použití testovania či refaktoringu, aj pri použití vzorov ide len o jeden z možných prostriedkov pre zlepšovanie kvality aplikácie, ktorý je potrebné využívať a ohľadom na špecifiká tímového projektu. Vhodné použitie zahŕňa poctivé premyslenie dôsledkov takéhoto konania.

## Schopnosti moderných nástrojov

Podľa [8] sa napriek výrazne zlepšujúcim sa nástrojom používaným pri implementácií a testovaní kvalita komerčného softvéru nezlepšila. Dôvodom je podľa autorov snaha o rýchly vývoj softvéru, pričom jeho komplexnosť neustále narastá.

Moja skúsenosť s používaním pokročilých IDE pri vývoji softvéru je prevažne pozitívna. Ide hlavne o zrýchlenie procesu implementácie, kedy funkcie akými sú dopĺňanie rozpísaných príkazov či automatické vygenerovanie časti často používaného kódu šetrí čas vďaka menšiemu počtu stlačení kláves. Navyše, tieto prostredia zvyčajne obsahujú samostatné moduly resp. integrované prostredia umožňujúce automatizované testovanie jednotlivých modulov a funkcií.

Ani najmodernejšie nástroje však nedokážu odhaliť logické chyby. Aj v prostredí tímového projektu teda platí, že i čiastočné nepochopenie modifikovanej časti kódu môže spôsobiť ťažko odhaliteľné logické chyby. Tímový projekt možno z hľadiska vhodnosti použitia moderných nástrojov tvorby softvéru porovnávať s komerčnými produktmi. Ak je pre používanú technológiu dostupné moderné IDE, je jeho použitie vo všeobecnosti prospešné a v konečnom dôsledku kvalitu produktu podľa môjho názoru nezníži. Vhodné je aj využitie jeho prípadných schopností v oblasti automatizovaného testovania, ktoré dokáže odhaliť chyby v skorých štádiách vývoja. V skoršom štádiu vývoja je samozrejme čas potrebný na odstránenie chyby rádovo menší. Automatizované testovanie zároveň dokáže odhaliť chyby vznikajúce v existujúcom kóde ako dôsledok pridania novej funkcie či modulu.

## Ako kvalitne vyriešiť tímový projekt?

Zabezpečovanie kvality softvérového projektu je proces, prebiehajúci počas celej doby jeho trvania. Spôsob použitia aktivít zahrnutých do procesu je však vždy potrebné prispôsobiť konkrétnym podmienkam danej organizácie či projektu. Pre kvalitné vyriešenie tímového projektu je teda nutné nielen použiť samotný proces zabezpečovania kvality, ale ho aj vhodne prispôsobiť všetkým jeho špecifikám. V konečnom dôsledku však celý proces závisí od samotných členov tímu, ich prístupu k nemu a teda k celému projektu.

# Použitá literatúra

1. BEAVER, M.J., SCHIAVANE, G.A.: The effects of development team skill on software product quality. In: *ACM SIGSOFT Software Engineering Notes*, vol. 31, 2006, no. 3, 1-5.

2. BERTOLINO, A.: Software Testing Research: Achievements, Challenges, Dreams. In: *2007 Future of Software Engineering*. International Conference on Software Engineering. IEEE Computer Society, Washington, 2007.
3. FELDMAN, S.: *Quality assurance: much more than testing*. Queue, vol. 3, 2005, no. 1, 26-29.
4. JIANG, Y., CUKI, B., MENZIES, T., BARTLOW, N.: Comparing Design and Code Metrics for Software Quality Prediction. In: *International Conference on software engineering.* Proceedings of the 4th international workshop on Predictor models in software engineering, ACM, Leipzig, 2008, 11-18.
5. JUNG, H.W., KIM, S.G., CHUNG, CH.S.: *Measuring Software Product Quality: A Survey of ISO/IEC 9126.* IEEE Software, vol. 21, 2004, no. 5, 88-92.
6. KHOMH, F., GUÉHÉNEUC, Y.G.: Do design patterns impact software quality positively?. In: *CSMR*, IEEE CS Press, 2008, 274-278.
7. STROGGYLOS, K., SPINELLIS, D.: Refactoring – Does it improve software quality?. In: *International Conference on Software Engineering.* Proceedings of the 5th International Workshop on Software Quality, 2007, 10
8. VAUGHAN-NICHOLS, S.J.: *Building better software with better tools.* Computer, vol. 36, 2003, no. 9, 12-14.
9. WARD, W.A., VENKATARAMAN, B.: Some observations on software quality. In: *ACM Southeast regional conference*, 1999, no. 2.

## Annotation

*How to produce quality team project?*

*Software quality is well defined and frequently used term. But what does software quality mean in the context of the team project at our faculty? In this essay I choose the best perspective on the quality of the software in the context of our team projects. The quality is influenced by every phase of the software development process from specification to deployment. However, probably the most important to quality is the design phase, which determines the effort needed in the subsequent phases of the lifecycle. Another important factor to mention is the overall knowledge and skills of the team members and their endeavour in the process of quality assurance which lasts through all lifecycle phases of the project. Frequent reasons for the decrease of the quality are late involvement of testing, misuse of refactoring and excessive reliance on design patterns or modern software development tools. In this essay I express my opinion on the proper use of the above-mentioned techniques, by which they are increasing the quality of the team project's result.*