CIEĽ: KVALITNÝ SOFTVÉROVÝ PRODUKT

"Nehovorte o chybách, tie budú hovoriť samé za seba."

Winston Churchill

Timotej Betina

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

Abstrakt. *Tvorbou a dodaním vysoko kvalitného softvéru staviame základ úspešného softvérového inžinierstva. Ako ale túto kvalitu dosiahnuť? Ciest ku kvalitnému softvéru je viacero. Dôležité je vybrať správny model pre tvorbu systému. V eseji som prediskutoval tri moderné existujúce modely, konkrétne RUP, MSF a XP. Porovnaním a skombinovaním ich vlastností získame ideálny model. Je to ale naozaj tak? Okrem výberu správneho modelu je taktiež dôležité zvoliť vhodný spôsob testovania, ktoré zvyšuje kvalitu softvéru. Testovanie je potrebné správne vybrať aj na základe zvoleného modelu. V eseji som opísal ideálne princípy testovania, na ktoré je v teoretickom svete zameraná pozornosť a snaha na ich docielenie. Dajú sa v skutočnom svete naozaj dosiahnuť? Nemalo by väčší význam zaoberať sa napríklad komunikáciou v tíme? Alebo na motiváciu vývojárov aby sa zamerali na kvalitný vývoj?*

Kľúčové slová: *model pre tvorbu systému, kvalita, RUP, MSF, XP, testovanie, komunikácia, motivácia*

Úvod

V súčasnosti je podstatne väčšia pozornosť venovaná kvalite softvérov, najmä s dôrazom na vytváranie kvalitných softvérových produktov. Softvérový vývoj je komplexný proces, ktorý integruje viaceré disciplíny a podieľajú sa na ňom mnohí analytici, dizajnéri

a programátori v priebehu niekoľkých rokov. Preto jednými z najdôležitejších vecí, aby boli tieto projekty úspešné, sú výber vhodného modelu pre proces softvérového vývoja a voľba vhodných testovacích metód. Okrem týchto technických rozhodnutí, ktoré musia tímy podieľajúce sa na vývoji vyriešiť, zostáva ešte stále v mnohých smeroch nedoriešená otázka komunikácie v rámci tímu a taktiež tímu so spolupracujúcim okolím. Táto esej sa preto bude zaoberať nielen výberom vhodného modelu a testovacích techník, ale najmä sa bude zameriavať na myšlienky, nápady a postrehy ako zlepšiť komunikáciu a motiváciu.

Kvalita

Čo je vlastne tá kvalita o ktorej sa toľko hovorí? Dá sa na ňu pozerať z viacerých uhlov pohľadu. Pre zákazníka je kvalitný ten výrobok, s používaním ktorého je spokojný a vydrží mu dlhý čas. Problémom je, že v dnešnej dobe čoraz menej firiem tvorí kvalitné produkty aj napriek tomu, že sa o kvalite toľko diskutuje a je na ňu zameraná výrazná pozornosť. Nie je to tým, že by to robili úmyselne, ale všetko dnes prebieha podstatne rýchlejšie, a preto aj firmy musia stíhať reagovať na to tempo. Dá sa teda hovoriť o určitom všeobecnom úpadku na kvalite. Hovorí sa, že výnimky potvrdzujú pravidlá a aj v tomto smere to platí, keďže sa stále dajú nájsť firmy, ktorých prvoradým cieľom je vytvárať kvalitné výrobky. Snaha o dosiahnutie kvality niekedy môže spôsobiť neskoršie uvedenie výrobku na trh a tým pádom spoločnosť nebude vnímaná ako líder. Zákazníci ale väčšinou vyššiu kvalitu vedia oceniť.

Kvalita v oblasti informačných technológií by mala byť vnímaná presne tak ako je vnímaná v menšinovom neinformatickom svete, čiže by sa mala brať ako priorita a mal by na ňu byť kladený hlavný dôraz. Je to totiž oblasť, v ktorej aj malé chyby môžu mať ďalekosiahle následky, a preto je dôležité aby jej bola venovaná nielen zvýšená pozornosť ale priam podriadený celý vývoj.

Ako dosiahnuť kvalitu?

Snaha o dosiahnutie kvality by mala byť automatická. Keď som robil rôzne projekty do školy, vždy som sa snažil na výsledok mojej práce pozrieť kriticky a kládol som si otázky: Je výsledný produkt kvalitný? Nedal by sa spraviť lepšie? Ako by sa dal vylepšiť? Okrem hľadania odpovedí na tieto otázky je tiež dôležité správne nastaviť kritériá, podľa ktorých hodnotím produkt. Ako príklad uvediem zákazníkov. Mnohokrát môže nastať nespokojnosť s výsledným produktom, čo ale neznamená, že ten produkt je zlý. Môže to vypovedať o tom, že zákazníci len nemajú jasne definované kritériá na kvalitu, pretože podľa môjho názoru, požiadavky na softvér a kritériá, podľa ktorých hodnotím produkt nemusia byť totožné, keďže spĺňanie všetkých požiadaviek neznamená, že sa so softvérom bude zákazníkovi dobre pracovať. Tým pádom je potrebné osobitne definovať obe tieto veci.

V minulosti sa nikdy nehovorilo toľko o kvalite ako dnes. Napriek tomu sa kvalita ako vlastnosť produktov brala automaticky. Nemyslelo sa na to, že je potrebné ju nejakým spôsobom explicitne testovať. Venovala sa dostatočná pozornosť samotnému výrobnému procesu a to do istého času stačilo. Postupne sa začali používať kontrolné prehliadky produktov, ktoré hľadali chyby na finálnych produktoch. Ďalším krokom bolo zvýšenie

kontroly výrobkov pri samotnom procese výroby. A posledným krokom vo vývoji je dôraz kladený na návrh takého procesu výroby, ktorý minimalizuje možné chyby produktu. Výsledok námahy o všeobecné zlepšovanie kvality bolo zavedenie sady noriem ISO 9000 [1]. Nástup týchto noriem mal výrazný prínos v jasnom zadefinovaní štandardu kvality ale súčasne zapríčinil aj určité problémy. Tie vznikajú najmä zlým pochopením noriem, keď organizácie dávajú medzi tieto normy a samotnú kvalitu produktu znamienko rovnosti, pričom si neuvedomujú, že tieto normy predstavujú len návod, odporúčanie ako vyrobiť kvalitný výrobok. Tým, že sú organizácie držitelia štandardov neznamená, že automaticky sú ich produkty kvalitnejšie.

Výber modelu

Okrem ISO noriem, skúsených ľudí podieľajúcich sa na vývoji a iných faktorov, vhodný a dobre fungujúci model vývoja softvéru je základ dosiahnutia kvalitného softvéru.

Tradičné modely (napr. vodopádový model) nestíhajú flexibilne reagovať na náročné požiadavky zákazníkov požadujúcich rýchlejší vývoj a nasadenie. Spoločné problémy týchto modelov z pohľadu kvality sú nasledovné [2]:

- Nesprávne porozumenie požiadavkám zákazníkov a neschopnosť flexibilne reagovať na zmeny požiadaviek.
- Neskoré odhalenie chýb softvéru. Súvisí to s tým, že sa väčšinou nezaoberajú explicitne kvalitou počas vývoja ale až pri testovaní. Dalo by sa povedať, že vodopádový model posúva riziko do budúcnosti a zahmlieva riziko až do vtedy, kým nie je neskoro niečo s tým robiť.
- Neberú procesy súvisiace s kvalitou ako súčasť samotného modelu, pričom tieto podporné procesy pre kvalitu sú veľmi dôležité z pohľadu dosiahnutia kvalitného produktu. Ak používajú nástroje na overovanie kvality, väčšinou sú to doplnky, čiže sú to určité praktiky overujúce softvér mimo samotného výrobného procesu.
- Zlá komunikácia medzi členmi tímu. Výsledok je, že sa ťažko určuje kto, čo, kedy a prečo zmenil.

Vlastnosti kvalitného modelu

Na základe predchádzajúcich problémov tradičných modelov boli identifikované určité body, ktoré by mal kvalitný model spĺňať [2]:

- Iteratívny vývoj. Pri takomto vývoji je identifikácia rizika doslova nanútená už v skorších fázach vývoja projektu. Taktiež určitá nezhoda medzi požiadavkami, návrhom a implementáciou je odhalená dostatočne skoro a záťaž na testovací tím je rovnomerne rozvrhnutá počas celého vývoja, nielen na konci.
- Identifikácia požiadaviek pri väčších projektoch je neustály proces a model musí integrovať tieto požiadavky do vývoja. Taktiež vývojári musia dokonale porozumieť zákazníkovi a jeho oblasti pôsobenia.
- Priebežné overovanie kvality testovaním počas celého vývoja je taktiež nevyhnutná vlastnosť kvalitného modelu. Význam priebežného testovania podľa

môjho názoru spočíva v tom, že ohodnotenie kvality projektu testami je objektívne, zakladajúce sa na výsledkoch daných testov, na rozdiel od subjektívneho ohodnotenia, ktoré sa zakladá napríklad na dokumente analýzy. Toto testovanie môže tiež odhaliť nezrovnalosti medzi požiadavkami, analýzou a implementáciou.
- Architektúra systému má taktiež výrazný podiel na kvalite výsledného produktu. Preto používanie vhodne navrhnutej architektúry uľahčuje integráciu produktu a jeho znovupoužitie v budúcnosti.
- V neposlednom rade, zvýšený dôraz musí byť kladený na tím ľudí podieľajúcich sa na vývoji. Dôležité je zvyšovanie motivácie prostredníctvom spolupráce a rozdelenie zodpovednosti na celý tím namiesto prenechania zodpovednosti na jedného člena, čiže nadriadeného.

Existujúce modely

V tejto časti sa pozrieme na tri asi najznámejšie a najpoužívanejšie modely v praxi, u ktorých je kvalita na prvom mieste [2].

RUP (Rational Unified Process)

Je najviac používaný model v priemysle. RUP spĺňa všetky vlastnosti, ktoré sme identifikovali v predchádzajúcej časti. Má veľmi dobre zadefinované overovanie kvality počas výrobného procesu vďaka tomu, že sú spúšťané pravidelné kontroly kvality na konci každej iterácie. RUP predstavuje kompletný framework (pozn. framework predstavuje komplexnú štruktúru prvkov, ktoré niečo umožňujú. V prípade RUP sú to jasne nadefinované prvky umožňujúce iteratívny vývoj softvéru s dôrazom na kvalitný vývoj.) a je veľmi dobre podporovaný softvérom Rational Software. V komplexnosti a zložitosti RUP tkvie ale aj nevýhoda, pretože porozumieť mu je veľmi komplikované. Aby ho mohli spoločnosti začať aplikovať v praxi musia zaplatiť nemalé peniaze na školenie ľudí, ktorí s ním budú pracovať. Ale bez dokonalého porozumenia samotnému procesu, podľa môjho názoru, nie je možné proces úspešne aplikovať.

MSF (Microsoft Solution Framework)

Podobne ako RUP aj MSF je komplexný framework. Taktiež podporuje všetky vlastnosti, o ktorých sme písali. Má ale jednu významnú prednosť pred ostatnými modelmi, a to, že kvalitu formálne definuje ako cieľ, resp. predmet samotného vývoja. To je vlastnosť, na ktorú mnohé iné modely zabúdajú aj napriek tomu, že takéto ponímanie kvality umožňuje jasnú definíciu kvality formou cieľov, čím máme zabezpečené, že sa kvalita stane vhodne merateľná a dosiahnuteľná vlastnosť. MSF sa taktiež silne orientuje na samotný tím.

RUP a MSF podľa môjho názoru určitým spôsobom redukujú samostatné myslenie. Ak totiž vývojári do bodky nasledujú predefinované procesy, prestávajú myslieť o najlepšom spôsobe ako vyriešiť pridelenú úlohu. Preto by možno nemali automaticky brať procesy, ktoré im určuje model ako fixné a nemenné, ale pokúsiť sa hľadať aj v nich slabé miesta a tie vylepšiť.

XP (Extreme Programming)

Tento model predstavuje agilný prístup k vývoju a na rozdiel od predchádzajúcich dvoch modelov nie je až tak formalizovaný. Kvôli tomu nie je úplne vhodný pre veľké projekty a zameriava sa najmä na vývoj menších produktov. XP úzko súvisí s vývojom riadeným testovaním (test-driven development) [3], ktorého základným princípom je to, že skôr ako programátor napíše nejakú funkciu, pripraví pre ňu test. To zabezpečí, že naprogramovaná funkcia bude spĺňať zadané požiadavky. Je to dosiahnuté najmä z dôvodu, že programátor ak by najprv klasicky napísal kód, je ovplyvnený tým, ako danú funkcionalitu navrhol a následne ju podľa toho bude aj testovať. Pri opačnom postupe práce sa táto chyba nemôže stať. Aj keď sa tento model vývoja nazýva riadený testom, treba si uvedomiť, že princíp nespočíva úplne v samotnom testovaní. Základným stavebným kameňom je skôr analýza a návrh a samotný test potom môže mať vplyv na to, aký kód programátor napíše.

Čo mne osobne najviac prekáža na XP je to, ako dostatočne motivovať programátorov aby venovali svoj čas písaniu testov ešte pred samotnou funkcionalitou. Pretože či si to pripúšťame alebo nie, od programátorov očakávame čo najrýchlejšie konečný výsledok a písanie testov ich na prvý pohľad od tohto cieľa vzďaľuje a vyžaduje si ich silnú disciplínu.

Niečo ale stále chýba

Podľa Grecheniga T., Heila S. a Zusera W. [2] je ideálny model taký, ktorý by mal tie vlastnosti, ktoré sú spoločné pre najpoužívanejšie modely. Čiže sú to vlastnosti, ktoré sme si vymenovali v časti *Vlastnosti kvalitného modelu*. Medzi týmito vlastnosťami figuruje aj to, že má byť kladený dôraz na tím. Dôležitosť tohto cieľa je podľa môjho názoru trochu v úzadí. Tím totiž musia tvoriť nielen vývojári, manažéri a testeri ale aj zákazníci a dokonca ešte aj klienti zákazníkov. Túto skutočnosť som si uvedomil pri mojej práci, kde sa podieľam na nasadení softvérového systému do prevádzky. Najviac podnetov a neporozumení systému prichádza práve od klientov našich zákazníkov aj napriek školeniu, ktoré absolvovali. Chyba bola v nedostatočnej komunikácií medzi týmito skupinami ľudí. Komunikácia so zákazníkmi a klientami zákazníkov je síce náročná ale netreba ju vnímať ako zásah do predstáv vývojárov. Vývojári si totižto mnohokrát myslia, že požiadavky zákazníkov im len nabúravajú vlastné plány, ale to nastáva okrem iného aj z dôvodu, že komunikácia prebieha v podstate len vtedy keď sa vyskytol nejaký problém. Dôležitá je neustála komunikácia, dokonca aj vtedy, keď všetko prebieha hladko.

Testovanie

Aby sme mohli o výslednom softvérovom produkte povedať, že je kvalitný, musíme ho najskôr podrobiť zaťažkávajúcim skúškam. Predtým, než sa začali v praxi používať modely, ktoré sme si opísali v predchádzajúcej časti, hlavná praktika kontroly kvality produktu bola testovanie. Treba si ale uvedomiť, že testovanie nezvyšuje kvalitu produktu, len sa ju snaží overiť. Momentálne testovanie zostáva stále na dôležitej pozícii, ale nie je už vyhradené iba na overovanie kvality finálneho produktu a používa sa počas celého vývoja.

Význam testovania spočíva v tom, že podrobne preskúmava softvér pri jeho priamej realizácii. Tým pádom poskytuje realistickú odozvu na správanie sa softvéru.

V teoretickom svete je vysoká pozornosť zameraná na *ideálne princípy testovania* [4], na ktoré sa aj my pozrieme v ďalšej časti tejto eseje.

Univerzálna testovacia teória

Vytvorenie takejto teórie je pretrvávajúci sen výskumníkov v oblasti testovania. Univerzálnou testovacou teóriou sa myslí jasne definovaný systém, o ktorý by sa mohli testeri opierať a taktiež by v porovnaní s ním videli silné a slabé stránky existujúcich testovacích techník. Ideálom je teda mať určitý mechanizmus, ktorý priradí efektívnu techniku testovania zvolenému cieľu, ktorý chceme testovaním dosiahnuť [4].

Ako cestu k dosiahnutiu tejto teórie osobne vidím v možnostiach využitia znalostného manažmentu. Konkrétne ontológie a dolovanie dát sa výborne hodia na uchovávanie a získavanie informácií z predošlých projektov a na základe nich voliť vhodný typ testovania. Dôležité pri takomto prístupe je pokúsiť sa získané poznatky ešte ďalej prispôsobiť na ten ktorý konkrétny projekt.

Model vychádzajúci z testovania

Predstava tohto princípu ideálneho testovania je taká, že namiesto klasického modelu, ktorého výsledok (čiže produkt) sa snažíme testovaním overiť, by sme sa mali orientovať smerom, kde v modeli priamo používame testovanie na dosiahnutie kvalitnejšieho softvéru [4]. V predchádzajúcej časti eseje sme si opísali modely na vývoj softvéru, ktoré už čiastočne tento ideálny princíp testovania spĺňajú. Problémom je, že každý jeden model má svoje chyby a nedá sa univerzálne používať. Preto je potrebné rozšíriť využitie znalostného manažmentu z predchádzajúcej časti na výber vhodného modelu a nielen testovania.

Plne automatické testovanie

V súčasnosti sa v softvérovom inžinierstve kladie veľký dôraz na automatizovanú produkciu softvéru, pričom sa využívajú moderné nástroje generujúce kvantá kódu s minimálnou námahou programátorov. O tomto prístupe sa dá polemizovať ale okrem výhod, najmä pre samotných vývojárov, prináša aj nevýhody, a to hlavne v tom, že testovanie nestíha úplne nadväzovať na tieto metódy vytvárania softvéru. Je teda dôležitá automatizácia testovania už či pokročilými technikami pomocou generovania vstupov alebo hľadaním nových inovatívnych procedúr testovacieho procesu. Sen je teda vytvoriť komplexný nástroj, ktorý sa sám postará o vytvorenie testovacích scenárov, vykoná ich a vyhodnotí [4].

Maximálna efektívnosť testovania

Tento cieľ sa zaraďuje medzi najväčšie sny výskumu o testovaní, ktorého veľká prekážka je narastajúca komplexnosť systémov, o ktorej sme si už hovorili aj pri predchádzajúcom cieli. Jeden zo smerov ako zlepšiť efektívnosť testovania, ktorým sa uberajú spoločnosti, je online testovanie, čiže sledovanie behu aplikácie pri reálnom nasadení. Asi najznámejším predstaviteľom je v súčasnosti Microsoft s produktom *Microsoft's Customer Experience*

Improvement Program [5]. Funguje to tak, že používateľ po chybe systému potvrdí odoslanie dát o danej chybe Microsoftu, ktorý s týmito informáciami následne pracuje. Problém je v tom, že ja ako používateľ takmer nikdy nepotvrdím tieto dialógové okná pýtajúce si dovolenie na odoslanie. Nemám totiž dôveru, že sa nepošlú aj dáta, ktoré by mohli byť zneužité. Ako alternatívu vidím vo vytvorení komunít, ktorých členovia by mohli priamo kategorizovať a zaznamenávať chyby systémov. Súčasťou týchto komunít by boli aj samotní vývojári systému. Problémom je, že väčšina členov z používateľskej obce by boli ľudia, ktorých zaujíma IT. Okrem týchto treba do komunít pritiahnuť aj menej informaticky gramotných používateľov, a to by mohlo byť docielené tak, že by im tí gramotnejší pomáhali s problémami a podobne. Tým by sa vytvoril systém ako zisťovať a monitorovať chyby počas priamej prevádzky systémov.

Nezabudli súdruhovia na niečo?

Teoretické sny výskumníkov majú určite svoj význam. Aj keď sa možno nikdy nedostanú do stavu, že si môžu spokojne pogratulovať za ich splnenie, dôležitý je aj samotný výskum a jeho medzivýsledky, ku ktorým sa dopracujú. Treba sa ale na to pozrieť aj z praktickejšieho pohľadu a porozmýšľať, či sa zameriavame na zodpovedanie správnych otázok.

Podľa môjho názoru je pri testovaní asi najdôležitejšia vec komunikácia. Hovorme teraz o aplikačnom testovaní. V súčasnosti to funguje tak, že zákazník komunikuje v podstate len s vývojárom. Problémom je, že vývojári po nejakom čase strávenom na projekte, keď sa už dostatočne rozrástol, strácajú prehľad a nedokážu sledovať všetky požiadavky a pripomienky zákazníkov. Svoju pozornosť zameriavajú skôr na dokumenty analýzy a podobne. Taktiež to môže nastať v prípade, ak zákazník nie je neustále prítomný pri návrhu každej časti systému. Testeri by preto podľa môjho názoru mali okrem detailného otestovania systému aj sledovať potreby zákazníka. Čiže ich komunikácia so zákazníkom by sa mala brať ako samozrejmá, keďže sa snažia čo najvernejšie napodobniť reálne použitie systému pri testovaní. Taktiež by im to mohlo pomôcť v tom, aby nemali tendenciu vytvárať testy, ktoré systém poľahky zvládne, keďže to sa väčšinou stáva vtedy, ak nemajú priamy kontakt s potrebami zákazníka.

Okrem komunikácie medzi testermi a zákazníkmi je dôležitá aj komunikácia medzi testermi a vývojármi. Často je pri takejto komunikácii potrebné riešiť situácie, keď tester vráti vývojárovi nejakú funkcionalitu s poznámkami, že to nefunguje tak ako má, pritom vývojár tvrdí, že daná funkcionalita vychádza z návrhu. V podstate majú pravdu obidve strany ale je zrejmé, že sa niekde stala chyba, či už pri špecifikácii požiadaviek alebo niekde inde. Vývojári majú podľa môjho názoru tendenciu podceňovať testerov a tým pádom môžu mať pocit, že pravda je na ich strane. Musia si ale uvedomiť, že testeri majú síce nie tak hlboký ale zato širší prehľad o problematike ako vývojári. Práve v takejto situácii by sa hodila komunikácia testerov so zákazníkmi z predchádzajúceho odstavca, čo by prinútilo vývojárov uznať nesprávne navrhnutie funkcionality.

Okrem komunikácie je veľmi dôležitá aj otázka motivácie samotných programátorov aby robili testy. Tohto problému sme sa už dotkli pri XP modeli, kde programátori sú nútení robiť testy pred samotnou funkcionalitou. Ako ich teda motivovať? Podľa môjho názoru existujú len dva hlavné motivátory. A to sú peniaze a čas. Keďže nedostatok času a plat sa pravdepodobne nezmenia, treba hľadať iné motivačné stratégie aj keď možno nie

až tak účinné. Jeden zo spôsobov je podľa mňa dôkladne oboznámiť programátorov s technikami, ktoré sa od nich vyžadujú a tým vzbudiť ich snahu správne používať zvolené techniky. Napríklad naučiť ich pracovať s unit testami pri agilnom programovaní a dôkladne s nimi otestovať efektívnosť práce pri písaní týchto testov oproti tomu, keď sa nepoužívajú. Ďalšia motivačná stratégia, ak tá predchádzajúca nezaberie, môže byť v autorite ich nadriadeného. Ak mu dôverujú, môže to aj ich vnútorne presvedčiť vo výhodnosť písania testov.

Záver

Ako sme si už povedali v úvode, cieľom tejto eseje bolo okrem objasnenia moderných prístupov k vývoju softvéru orientujúceho sa na kvalitu a ideálnymi princípmi testovania najmä upriamenie pozornosti na komunikáciu v tíme. Tá dokáže výrazne nepriamo zvýšiť kvalitu softvéru. Už či lepšou atmosférou v tíme alebo zrozumiteľnejšie definovanými požiadavkami zákazníkov. Je preto potrebné aby sme si aj my, tím študentov na predmete Tímový projekt, porozumeli a pokúsili sa zvýšiť kvalitu výsledku našej práce realizovaním myšlienok, ktoré sú opísané v tejto eseji.

Použitá literatúra

1. GILL, S. N. Factors Affecting Effective Software Quality Management Revisited. In *ACM SIGSOFT Software Engineering Notes* [online]. March 2005, vol. 30 [cit. 2010-10-17]. Dostupné na internete: <http://portal.acm.org/citation.cfm?id=1050849.1050862>. ISSN:0163-5948.
2. GRECHING, T., HEIL, S., ZUSER, W. Software Quality Development and Assurance in RUP, MSF and XP - A Comparative Study. In conference *International Conference on Software Engineering* [online]. 2005 [cit. 2010-10-17]. Dostupné na internete: <http://portal.acm.org/citation.cfm?id=1083300>. ISBN:1-59593-122-8.
3. JANZEN, D., SAIEDIAN, H. Test-Driven Development: Concepts, Taxonomy, and Future Direction. In *Computer* [online]. September 2005, vol. 38, no. 9 [cit. 2010-10-17], pp. 43-50. Dostupné na internete: <http://www.computer.org/portal/web/csdl/doi/10.1109/MC.2005.314>.
4. BERTOLINO, A. Software Testing Research: Achievements, Challenges, Dreams. In conference *International Conference on Software Engineering* [online]. 2007 [cit. 2010-10-17], pp. 85-103. Dostupné na internete: <http://portal.acm.org/citation.cfm?id=1253532.1254712>. ISBN:0-7695-2829-5.
5. Oficiálna stránka Microsoft [online]. [cit. 2010-10-17]. Dostupné na internete: <http://www.microsoft.com/products/ceip/EN-US/default.mspx>.

Annotation

Goal: High-quality software product

The ground of successful software engineering is built on creating and delivering high quality software. But how is this quality achieved? There are many roads to quality software. One of the most important tasks is to choose the right software development process model. In this essay, I discussed three modern existing models, namely RUP, MSF and XP. Ideal model is obtained by comparing and combining practices of these models. But is this really true? Choosing the right testing methods is also important and depends on selected model. Testing increases the quality of software. In this essay I described the ideal principles of testing. Theoretical world focuses attention on these principles and spends significant effort to achieve them. Are they really achievable in real world? Wouldn't be communication in team more important? Or wouldn't motivation of developers to focus on quality software development make more sense?