Zabezpečenie kvality pri vývoji softvéru a testovanie

FILIP KOMOROVSKÝ

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

Abstrakt. Zabezpečenie kvality softvéru je plánované a systematické zloženie aktivít ktoré zaručujú, že softvérové procesy a produkty budú spĺňať dané požiadavky, štandardy a iné špecifikované procedúry. Procesy zahŕňajú všetky aktivity počas dizajnu, vývoja, vylepšovania a udržiavania softvéru. Produkty predstavujú softvér, s ním spojené dáta, dokumentáciu a inú podporu. Na zabezpečenie kvalitu je dôležité myslieť počas celého životného cyklu softvéru, preto sa pokúsim v tomto článku porovnať niektoré vybrané metodiky vývoja softvéru (XP, MSF, ...) z hľadiska zabezpečenia kvality, a rozobrať vývoj softvéru riadený pomocou testov (tzv. Test driven development).

Úvod

Zabezpečovanie kvality softvéru je prirovnávané ku zabezpečovaniu kvality pri normálnej výrobe. No treba dávať pozor, pretože medzi softvérovým produktom a inými výrobkami sú značné rozdiely. Hlavný a najdôležitejší rozdiel je v tom, že softvérový produkt nie je viditeľný, tak ako obyčajný hmatateľný výrobok. Preto nie je jednoduché ohodnotiť jeho cenu, funkcionalitu a správnosť tak jednoducho ako pri iných výrobkoch. Čo je ale dôležitejšie, radový výrobok, po tom ako sa dokončí jeho výrobný proces, je pokladaný za hotový a pripravený na predaj. Ale softvérový produkt nie, neustále sa vyvíja, dotvára a prispôsobuje novým požiadavkám a podmienkam. Preto procesy a metódy ktorými manažujeme, monitorujeme a meriame kvalitu softvéru sú pretrvávajúce, plynulé a hlavne musia obaľovať celý životný cyklus softvérového produktu.[3]

Výroba softvéru stále viac a viac rastie a naberá na komplexnosti. Softvérový produkt je väčšinou vytváraný pre istú ohraničenú cieľovú skupinu ľudí. Toto je tiež jeden z faktorov na ktorý musia vývojári hľadieť a pri návrhu ho zohľadňovať. Čiže výsledný produkt musí byť akceptovaný hlavne cieľovou skupinou na ktorú je orientovaný. Preto prichádza do hry softvérové testovanie. Testovanie nie je len hľadanie chýb a nezrovnalostí, je to kompletná disciplína na meranie a zabezpečovanie

Manažment v softvérovom inžinierstve, október 2007, s. 1-1.

kvality softvérového produktu. Niekedy sa testovanie označuje aj ako kontrolovanie kvality.

Zabezpečenie kvality

Čo sa podieľa na zabezpečení kvality? Samozrejme hlavne testovanie. No treba dodať, že kvalita sa na produkte nedá testovať. Solídny plán by mal chyby odchytiť a dať odhad kvality. Dobrý plán zaistí, že dizajn aj implementácia sú správne a že produkt spĺňa všetky požiadavky pred vydaním. No výborný plán zabezpečenia kvality je pokročilá organizácia, ktorá v sebe má analýzu chýb a kontinuálne vylepšovanie (Tento cyklus je charakteristický pre vyspelé organizácie).

Pre fyzický produkt, zabezpečovanie kvality predstavuje kontrolu procesu výroby, kontrolu dizajnu, plány testov, štatistické metódy a veľa iného. Pri slabšom využití, existujú akurát občasné monitorovania výroby. Poprípade sú niektoré kusy náhodne testované na rôznych úsekoch. V extrémnych prípadoch, je každý krok monitorovaný a zaznamenávaný, medziprodukty sú testované do najmenších detailov a veľa výsledných produktov sa nakoniec pri testovaní aj zničí. Len niektoré produkty sa nakoniec dostanú do predaja.

Pre softvérové produkty existuje však mnoho procesov zabezpečovania kvality na výber. Závisí od štruktúry organizácie, dôležitosti softvéru, rizík a ceny neúspechu. Toto sú body ktoré by sa mali neustále a periodicky prehodnocovať. A na základne výsledkov by sa mali vytvárať a upresňovať ďalšie postupy.[2]

Prístupy k zabezpečeniu kvality

Keďže zabezpečovanie kvality je proces, je prirodzené, že s ním budú spojené špeciálne role a organizácie. V niektorých firmách je normálne, že sa o kvalitu starajú ľudia z vývoja, rovnako ako sa starajú o testovanie a návrh. Toto môže platiť v menších softvérových firmách, no vo všeobecnosti sa to neujme. Preto sa vo veľkých firmách o zabezpečenie kvality starajú špecializované tímy ľudí, ktoré sú oddelené od vývojovej časti. Tieto tímy sú zodpovedné za proces vývoja a určujú detaily ďalšieho napredovania. Nie je dôležité, aby boli tímy extrémne veľké nato aby boli efektívne. Podstatné je, aby boli zložené zo skúsených ľudí, ktorí majú výborné vedomosti o vytváranom produkte.[2]

V ďalšej časti sa budem podrobnejšie venovať jednému vybranému modelu vývoja softvéru. Pôjde o model Extreme programming. Pokúsim sa vyzdvihnúť body dôležité z hľadiska zabezpečovania kvality softvérového produktu.

Extreme programming

Extrémne programovanie (eXtreme programming, XP) je jeden z modelov vývoja softvéru. Je to súhrn pravidiel a postupov ktoré stelesňujú jednotlivé hodnoty dané touto metódou. Zástanci tohto modelu veria, že dodržiavanie týchto pravidiel a postupov (postupov ktoré sú dovedené do tzv. extrémnych úrovní) vedie k vývojovému procesu ktorý je citlivejší, alebo lepšie povedané agilnejší, k potrebám a špecifikáciám používateľa, a vo výsledku vedie, a to nás zaujíma najviac, ku kvalitnejšiemu softvérovému produktu.[1]

Extrémne programovanie je opisované ako:

- Pokus o spojenie ľudskosti a produktivity
- Cesta k zlepšeniu
- Štýl vývoja
- Disciplína softvérového vývoja

Jedným z hlavných cieľov je zníženie ceny pri zmene. V tradičných metódach vývoja (ako napr. SSADM) sú požiadavky pre systém určené na začiatku vývoja projektu. To znamená ale, že akékoľvek úpravy počas vývoja sú veľmi náročné a drahé. XP je cielené tak, aby tieto náklady znížilo využitím základných hodnôt a princípov. Projekt by mal tak byť flexibilnejší a nemal by mať veľké problémy s prispôsobením sa ku zmenám. Podľa môjho názoru, rýchle adaptovanie sa na zmeny je jeden z kľúčových bodov pri zabezpečení kvality. Kvalita ako taká sa nedá dosiahnuť, pokiaľ neustále zmeny a zasahovanie do procesu vývoja narušujú stabilné napredovanie ku finálnemu kvalitnému produktu. Zmeny a úpravy sú samozrejme vo veľa prípadoch opodstatnené a možno aj nevyhnutné, no ako platí všade, extrémy nie sú zdravé. A preto treba hľadať vyvážený stav.

Medzi niektoré základné hodnoty XP patria napríklad:

- Komunikácia
- Jednoduchosť
- Odozva

Vytváranie akéhokoľvek softvéru vyžaduje častú komunikáciu a upresňovanie rôznych vecí medzi členmi týmu. Každý člen by teda mal mať jednoduchý náhľad na celkový systém a jeho obraz zo strany používateľa. Preto aj jedna z hodnôt XP je práve neustála vzájomná komunikácia v týme. Takto sa ľahko predchádza veľkému množstvu chýb ktoré vznikajú zo vzájomného nepochopenia sa medzi vývojármi. Z praxe môžem povedať, že najčastejšie chyby vznikajú z nedorozumení, alebo podcenenia častých konzultácii medzi členmi týmu.

Čo sa jednoduchosti týka, ide tu hlavne o to, aby sa pri vývoju prizeralo skôr na tú stránku a funkcionalitu ktorá je potrebná teraz, a teda bola zachovaná jednoduchosť a zrozumiteľnosť softvéru. A nie teda riešiť zložitý systém ktorý bude využívať budúce technológie. Dôležité je to čo je teraz, a pokiaľ bude zachovaný jednoduchý a kvalitný dizajn, nové technológie bude jednoduché dopracovať v ten správny čas.

Na odozvu sa dá pozerať z 3 základných hľadísk.

1. **Odozva od systému** – pripravovaním čiastkových testov je jednoduché a zistiť ako sa správa celý systém po implementovaní jednotlivých zmien
2. **Odozva od zákazníka** – sem spadá akceptačné testovanie ktoré je vytvárané zákazníkom a testovacím týmom. Zákazník takto môže v pravidelných intervaloch vidieť akým smerom a rýchlosťou napreduje vývoj, a poprípade ho usmerniť správnou cestou.
3. **Tímová odozva** – tím dokáže stanoviť prostriedky potrebné pri nových zmenách a požiadavkách zákazníka

Základné princípy vychádzajú zo spomínaných hodnôt. Odozva je najužitočnejšia ak je využívaná často, takto sa dá reagovať na zmeny rýchlo a nie keď je už neskoro. Takisto sa dá usmerňovať vývoj tým správnym smerom. V tomto najviac pomáhajú už spomínané čiastkové akceptačné testy.

XP definuje aj základné aktivity ktoré by mali byť dodržované pri procese vývoja. Za najdôležitejšiu vec, ktorá ma jedinú pravú výpovednú hodnotu, označujú zástanci samotný kód a teda *programovanie*. Bez neho nie je nič. Pomocou priameho naprogramovania jednotlivých možných riešení nejakého čiastkového problému možno napríklad vybrať najrozumnejšie a najlepšie riešenie. Zdrojový kód takisto pomáha pri komunikácii v týme, keď je napríklad jednoduchšie ukázať kolegovi časť kódu na ktorej môže lepšie pochopiť o čo sa jedná, ako mu to vysvetľovať slovne. Z touto aktivitou úzko súvisí aj tá nasledujúca, a tou je *testovanie*. Bez testovania si nemôže byť nikto istý či je daná časť kódu alebo návrhu správna. Ďalšou aktivitou je *počúvanie*. Programátor málokedy vie o systéme alebo produkte na ktorom sa podieľa viac, ako je ohraničené jeho časťou kódu, resp. častou za ktorú je priamo zodpovedný. No mal by vedieť viac, mal by mať o systéme väčší obraz a vedieť aké sú priame požiadavky zákazníka. Ide teda o to aby bol spojenie zákazníka s programátorom resp. vývojárom čo najpriamejšie. Poslednou aktivitou je *vytváranie dizajnu*. Táto aktivita samozrejme nemôže byt opomenutá, keďže jednotlivé časti aplikácia musia byť programované vzhľadom na istú celkovú logiku, a musia spĺňať dané výstupy a vstupy aby bola komunikácia medzi nimi bezchybná.

Porovnanie vlastností modelov vývoja softvéru

V tejto časti sa pokúsim porovnať XP s dvoma ďalšími (Microsoft solution framework a Rational unified process) podobnými metódami vývoja softvéru z hľadiska zabezpečenia kvality.

Zabezpečovanie kvality pri MSF

MSF (Microsoft solution framework) začína na počiatku projektu definovaním základných požiadaviek a potrieb. Zakladá si na zabezpečení kvality asi najviac zo spomínaných troch metód. Zaručuje základnú úroveň kvality, aj keď jej metodika môže byť upravená vzhľadom na potreby, pretože kľúčové elementy systému zostávajú nemenné. Kvalita je tu definovaná ako jasný a dôležitý cieľ, toto je jeden z bodov v ktorom sa MSF odlišuje od ostatných. Ďalšou silnou stránkou je zameranie na tímy, používa sa unikátny prístup pri ktorom sú pre každú rolu definované jej ciele. Nakoniec manažment rizík je reprezentovaný vlastným modelom, ktorý predstavuje jeden z najlepších modelov manažmentu rizík v procese vývoja softvérových produktov.[4]

Zabezpečovanie kvality v RUP

RUP (Rational unified process) nemá definovanú kvalitu ako jeden z cieľov, tak ako MSF. No zakladá si na silnom overovaní kvality počas celého procesu vývoja. Po každej vývojovej iterácii je spustená jasne definovaná procedúra na verifikovanie a otestovanie dosiahnutých cieľov, poprípade cieľov ktoré neboli zvládnuté. Samozrejme nachádza v tomto prípade aj zdroje chýb, prečo sa ten ktorý cieľ nepodarilo splniť. Keďže splnenie cieľov pri jednej iterácii je kľúčové pre spustenie ďalšej iterácie, kde tieto ciele predstavujú očakávané vstupy. RUP rovnako ako MSF je tiež silne zamerané na tímovú prácu. Nie teda na programovanie v pároch, ktoré ale samozrejme môže byť do procesu zapracované.[4]

Zabezpečovanie kvality v XP

XP (eXtreme programming) je v prvom rade zamerané na menšie projekty, programovanie v pároch a vývoj úzko spätý so zákazníkom. Zákazník hrá kľúčovú rolu pri vývoji, a priamo dohliada na výsledok po krátkych a častých iteráciách. Takéto časté akceptačné testovanie zvyšuje podstatne výslednú kvalitu produktu, ako aj celého vývoja. Najlepšie sa javí XP, keď sa zapracuje do vyššie spomínaných metód (MSF alebo RUP).[4]

Vývoj riadený testami

Nakoniec by som chcel spomenúť aj jednu techniku vývoja softvéru ktorá, podľa mňa, môže veľkou mierou prispieť ku kvalitnejšiemu softvéru. Ide o tzv. vývoj riadený testami (Test-driven development, TDD). Je technika vývoja softvéru, pozostávajúca z krátkych vývojových iterácii. Každá iterácia pozostáva z testu a implementácie funkcionality ktorá bude daným testom overovaná. Pred implementovaním funkcionality v programe sa najprv vytvorí súbor testov, a až potom sa implementuje. Takto sa rapídne zvyšuje odozva po každej zmene v kóde.

Postup pri vývoji softvéru pomocou tejto techniky vyzerá nasledovne:

- **Vytvorenie testu -** Na začiatku sa vytvorí test na danú časť problému ktorá sa bude implementovať. Tým pádom vývojár ktorý bude test navrhovať musí mať úplný prehľad o požiadavkách a špecifikácii daného problému. Tým pádom sa na požiadavky prihliada pre písaním kódu, a nie až po tom.
- **Spustenie testu nad existujúcou časťou kódu a zistenie chyby -** V tomto momente sa zistí, či bol test dobre navrhnutý. A to tak, že existujúci kód testom neprejde. Pretože ani nesmie. Takto sa eliminuje možnosť, že by testom prešiel aj starší kód, a tým pádom by test nemal žiadnu výpovedný hodnotu.
- **Implementovanie nového kódu -** Teraz sa až implementuje časť kódu ktorá má prejsť navrhnutým kódom. Je dôležité aby naozaj bola implementovaná len časť pre ktorú bol napísaný kód, a nie ďalšia funkcionalita. Implementácia nemusí byť dokonalá, dôležité je aby zbehol test. V neskoršej časti sa bude kód upratovať a optimalizovať.
- **Spustenie testu s pozitívnym výsledkom -** Spustí sa test, a implementovaný kód prejde s očakávaným pozitívnym výsledkom.
- **Upratovanie kódu -** Nakoniec cyklu sa kód uprace, je samozrejme možné ďalej priebežne testovať kód nad predchádzajúcimi testami aby sa predišlo chybám pri upratovaní.

Takto teda vyzerá jeden cyklus pri vývoji pomocou TDD. Táto technika má veľa pozitívnych vlastností. Jednou z nich je napríklad to, že programátor pri vývoji, a naprogramovaní ďalšej funkcionality, na ktorej pripravený test nečakane spadne, nemusí prácne zisťovať kde nastala chyba. Ale môže sa jednoducho vrátiť k pôvodnej implementácii ktorá zbehla na poslednom teste, a skúsiť iný postup implementácie.

Ďalšou výhodou je, že chyby sa v kóde odhalia a opravia už v ich zárodku, po prvom testovaní danej funkčnej časti. Pretože testy sú stavané aj z používateľského hľadiska a tým pádom sa eliminujú chyby ktoré by sa pri iných metódach objavili až pri záverečnom testovaní. Priebežne tento systém vedie k väčšiemu objemu kódu, ale

vo výsledku, po jednotlivých úpravách je kód kvalitnejší a kompaktnejší. Všetko toto úsilie nakoniec prispieva ku zníženiu výskytu chýb a tým pádom aj ku kvalitnejšiemu softvéru.

Záver

Porovnaním 3 vybraných modelov procesu vývoja softvéru z hľadiska kvality som sa presvedčil, že majú veľa spoločného. Aj keď sú zreteľné rozdiely, ktoré vyplývajú z cieľového využitia. XP je napríklad orientované skôr na menšie tímy, programovanie vo dvojici a podobne, no kvalitou za MSF ani RUP určite nezaostáva. Preto som si aj tento model vybral na podrobnejšiu analýzu, keďže by som si vedel predstaviť ako by sa dal využiť pri našom školskom tímovom projekte. Takisto aj TDD je technika, ktorá výrazne napomáha ku kvalite vývoja softvérového produktu, by sa mohla stať jednou z možností pri vývoji nášho tímového projektu.

Použitá literatúra

1. Extreme Programming: http://www.extremeprogramming.org
2. Feldman S.: *Quality Assurance: Much more than testing,* ACM Queue, Feb. 2005
3. Software Quality: http://www.swebok.org/ch11.html
4. Zuser W., Heil, S., Grechenig T.: *Software Quality Development and Assurance in RUP, MSF and XP – A Comparative Study,* ACM, 2005, 2-5.

Annotation

Software quality assurance and testing

Software assurance is the planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures. Processes include all of the activities involved in designing, developing, enhancing, and maintaining software. Products include the software, associated data, its documentation, and all supporting and reporting paperwork. It's very important to think about quality assurance through complete life cycle of software. In this article I will try to compare 3 selected methodologies for software developing (XP, MSF, and RUP) from quality assurance point of view. Than I will try to analyze test-driven software development.