Ako nájsť vhodný systém zabezpečenia kvality pre konkrétny softvérový projekt?

STANISLAVA LEITMANOVÁ

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
leitmanova@wtools3d.com

Abstrakt. V súčasnej dobe dochádza k čoraz väčšej informatizácii našej spoločnosti. Používanie softvéru pomaly preniká do takmer všetkých oblastí nášho života a je jeho každodennou súčasťou. Prevažná väčšina firiem sa stáva viac-menej závislá od používania rôznych softvérových nástrojov. Z tohto dôvodu rastie aj dôležitosť kvality softvérových systémov. Téma kvality softvérových systémov je v dnešnej dobe veľmi diskutovaná a existuje množstvo postupov, modelov a metodík , ako zabezpečovať manažment kvality. Medzi softvérovými projektmi je veľmi veľká variabilita a každý z nich môže mať iné požiadavky na kvalitu. Preto je veľmi dôležité vybrať vhodný systém riadenia kvality v závislosti od povahy projektu, veľkosti tímu a ďalších atribútov.V tejto práci sa zameriam hlavne na zabezpečovanie kvality v malých softvérových projektoch, podobných tým, ktoré vznikajú v rámci predmetov na našej fakulte.

Čo znamená kvalita v softvérovom projekte?

Problematika zabezpečovania, riadenia a kontroly kvality pri softvérových projektoch je omnoho zložitejšia ako pri bežných výrobkoch a službách. Už samotné definovanie kvality pre softvérové systémy je veľmi nejednoznačné.

Jedna zo základných a všeobecných definícií kvality hovorí, že kvalita je miera uspokojenia potrieb zákazníka Softvérový produkt ako taký je však nehmotný "výrobok" a neexistujú presne definované kritériá či postupy pre hodnotenie jeho kvality. Medzi softvérovými projektmi je veľká variabilita, preto aj požiadavky na ich kvalitu sú rôzne a vyplývajú z povahy projektu. Ďalším problémom pri posudzovaní kvality je, že požiadavky na softvér sú často nekompletné a nekonzistentné. Z tohto dôvodu nie je jednoduché špecifikovať určité konkrétne požiadavky na kvalitu pre daný softvérový systém.

Preto existujú rôzne prístupy a pohľady na manažment kvality v softvérových systémoch. Pri posudzovaní kvality softvéru rozoznávame päť rôznych definícií[1]:

Manažment v softvérovom inžinierstve, október 2007, s. 1-6.

1. Produktová kvalita - presné a merateľné premenné ako vlastné charakteristiky produktu (objektívne atribúty kvality)
2. Procesná (výrobná) kvalita - súlad so špecifikáciami určenými v predpisoch, štandardoch a normách
3. Užívateľská kvalita – použiteľnosť, uspokojovanie požiadaviek zákazníka
4. Hodnotová kvalita - kvalita sa vzťahuje ku nákladom (efektivita a výkon za prijateľnú cenu a náklady)
5. Transcendentná - kvalita je absolútna a univerzálne rozpoznateľná hoci nie je možné ju definovať presne. Len skúsenosti môžu naučiť rozpoznávať kvalitu

Hodnotenie kvality každého softvérového systému môžeme rozdeliť do troch úrovní:

- Miery vnútornej kvality (internal quality), ktoré slúžia ako predikátory budúcej kvality v dobe vývoja produktu.
- Miery vonkajšej kvality (external quality), ktoré merajú atribúty, ktoré spĺňajú požiadavky a potreby používateľa.
- Miery kvality použitia (quality in use), ktoré merajú kvalitu procesu využívania produktu a prínos využívania pre koncového používateľa.

Toto hodnotenie úzko súvisí so životným cyklom vývoja softvéru. Napríklad v dobe vývoja softvéru môžeme hodnotiť jedine jeho vnútornú kvalitu. Táto kvalita však nespĺňa žiadne potreby používateľa preto v skutočnosti ani nie je kvalitou podľa svojej pôvodnej definície. Avšak napriek tomu je zrejmé, že vnútorná kvalita je nevyhnutným predpokladom kvality vonkajšej a kvality použitia. Čím skôr v procese vývoja softvéru zistíme prípadný nedostatok, či ohrozenie kvality, tým je jeho náprava lacnejšia.

Manažment kvality – prístupy a metódy

Manažment kvality sa zameriava na dosahovanie požadovanej úrovne kvality pomocou definovania vhodných štandardov a procedúr a sledovania ich dodržiavania.
Je to koordinácia činností v rámci podniku či tímu, ktoré smerujú k zabezpečeniu kvality. Realizuje sa prostredníctvom konkrétneho systému manažmentu kvality.
S manažmentom kvality súvisia tieto aktivity: zabezpečenie kvality, plánovanie kvality, riadenie/kontrola kvality.

Existujú rôzne normy a postupy pre manažment kvality. Ako bolo spomenuté v úvode, kvalita softvérového produktu je veľmi ťažko merateľná. Preto sa často používa prístup systémového riadenia kvality. Tento prístup je charakteristický predpokladom, že ak sú kvalitné procesy návrhu a výroby, bude kvalitný aj produkt. Preto vzniklo aj viac metodík pre hodnotenie softvérových procesov, ktoré sú založené na tomto prístupe. Najznámejšia z týchto metodík, používaná práve pre firmy zaoberajúce sa tvorbou softvéru na zákazku, je CMM (Capability Maturity Model).

Myslím si, že výber metodiky riadenia kvality zohráva pri vývoji veľmi dôležitú úlohu a pri tomto výbere treba vychádzať z povahy a požiadaviek na softvérový produkt. Je dôležité si uvedomiť, že nie každá metodika je vhodná pre každý softvérový produkt. Napríklad zavedenie komplexného systému manažmentu kvality, ktorý vychádza z určitej normy, môže byť veľmi prospešné pre väčšiu firmu, ktorá dodáva softvérové produkty na zákazku. Výhodou týchto noriem je možnosť získania certifikácie, ktorá dáva firme určitý „kredit" a takáto firma potom pre zákazníka pôsobí vierohodnejšie ako napríklad konkurenčná firma, ktorá žiaden certifikát nemá. Na druhej strane, pre menšie tímy alebo rôzne nekomerčné či výskumné projekty by bol takýto manažment kvality zbytočne komplikovaný a určite by v takom projekte pôsobil skôr kontraproduktívne.

Práve projekt, ktorému sa venujeme na predmete Tvorba softvérového systému v tíme, je dobrým príkladom projektu, kde nie je vhodné použiť nejaký komplexný prístup k riadeniu kvality. Je to projekt zaoberajúci sa simulovaným robotickým futbalom. Je to časť umelej inteligencie, ktorá sa neustále vyvíja a týmto sa stále zvyšujú a dopĺňajú aj požiadavky na takýto typ softvérového projektu. Viac ako o softvérovom projekte by sme o ňom mohli hovoriť ako o výskumnom projekte, pretože samotná implementácia nie je jeho jediným cieľom. Jeho hlavný prínos je práve vo výskume a „objavovaní" nových algoritmov či prístupov k riešeniu. Z tohto dôvodu je naozaj obtiažne definovať požiadavky na kvalitu takého projektu.

To, aký prístup ku kvalite si zvolíme v konkrétnom projekte, je sčasti dané tým, akú metodiku vývoja softvéru preferujeme. V súčasnosti existujú dva hlavné prúdy metodických prístupov, ktoré sa označujú ako klasické metodiky a agilné metodiky.

Klasické metodiky vychádzajú z presvedčenia, že vývoj softvéru sa dá popísať, plánovať, riadiť a merať. Snažia sa podrobne definovať procesy a činnosti. Sú založené na sériovom – vodopádovom modeli vývoja softvéru. Pri tomto spôsobe prebiehajú jednotlivé fázy životného cyklu projektu, ako analýza, návrh, implementácia, zavedenie do prevádzky, sekvenčne za sebou. Existujú však aj klasické metodiky založené na iteratívnom vývoji. Iteratívny vývoj je taký, kde jednotlivé fázy vývoja prebiehajú opakovane a výsledkom každej iterácie je funkčná verzia systému.

Agilné metodiky sú založené na myšlienke, že najlepší spôsob ako preveriť správnosť navrhnutého systému je vyvinúť systém alebo jeho časť čo najrýchlejšie, predložiť ho zákazníkovi a na základe spätnej väzby upraviť. Agilné metodiky je výhodné použiť najmä pri projektoch, kde poznáme dopredu len hrubé požiadavky a nevieme popísať softvérové procesy. Preto sa hodia najmä na rôzne výskumné projkety prípadne pre menšie tímy. Sú zamerané na vývoj nového riešenia, nových prístupov a algoritmov.

Jedna z metodík, ktoré patria k agilným metódam vývoja softvéru je vývoj riadený testami.

Vývoj riadený testami

Vývoj riadený testami – Test Driven Development je technika vývoja softvéru, ktorá patrí medzi agilné metódy. Pri tejto metóde tvorba testov predchádza tvorbe samotného

kódu. Cieľom vývoja riadeného testami je špecializácia a nie validácia. Tento spôsob písania testov núti programátora sa zamyslieť nad návrhom pred samotnou implementáciou. Základný cyklus testami riadeného vývoja možno opísať nasledujúcimi krokmi[3]:

1. Pridanie testu: Prvý krok predstavuje pridanie nového testu, v podstate práve dostatok kódu na to, aby test zlyhal. V testami riadenom vývoji každá nová vyvíjaná funkcia začína písaním testu. Novovytváraný test obsahuje práve minimum kódu potrebného na to, aby test zlyhal. Tento test musí zlyhať, pretože je napísaný ešte pred samotnou implementáciu danej časti. Pri podobností funkcií je možné použiť a modifikovať existujúci test.
2. Spustenie všetkých testov - spustenie testov v tomto kroku slúži k overeniu zlyhania novo pridávaného testu. Týmto sa overí, že zvyšok testov prejde úspešne a nový test neskončí úspešne bez požadovania nového kódu.
3. Napísanie nejakého kódu - tento ďalší krok spočíva v napísaní nejakého kódu postačujúceho k tomu, aby test prešiel. Tento "nejaký" kód môže byť akokoľvek nedokonalý a neelegantný, pretože neskoršie kroky ho vylepšia a zdokonalia. Je veľmi dôležité, aby bol funkčný kód navrhnutý jedine k splneniu požiadaviek testu. Žiadna ďalšia funkcionalita, ktorá nie je definovaná testom, by nemala byť implementovaná.
4. Spustenie automatizovaných testov - toto spúšťanie testov je principiálne odlišné od spúšťania testov v druhom kroku. Kým pri spúšťaní testov v druhom kroku ide o to, aby programátor videl, že práve pridaný test zlyhá, spúšťanie testov v štvrtom kroku predstavuje overenie funkčnosti kódu. Ak testy zlyhajú, programátor musí opakovane upravovať kód a následne púšťať testy, kým nedopadnú úspešne
5. Refaktorovanie kódu - refaktorovanie znamená každú zmenu kódu, ktorá prispeje k lepšej čitateľnosti, alebo zlepšeniu štruktúry bez zmeny funkcionality. Cieľom refaktorovania je čistý kód, ktorý funguje. V tejto fáze sa kód prečisťuje podľa potreby. Opakovaným spúšťaním testov pomedzi refaktorovanie má vývojár možnosť kontrolovať dodržanie funkcionality refaktorovaného kódu. Kód sa opakovane refaktoruje pokým nie je dostatočne „čistý" v závislosti od požiadaviek.

Myslím, že vývoj riadený testami má priamo vo svojom princípe „zabudované" isté zabezpečenie kvality. Je to práve z toho dôvodu, že po každom pridaní nejakej novej funkcionality sa hneď program testuje. Uskutočňovaním veľkého množstva testov takto predchádzame mnohým chybám a takmer úplne odpadá hľadanie chýb pomocou debuggovania. Nielen že chyby odhalíme, ale odhalíme ich aj včas, čo je tiež veľmi dôležité pri vývoji softvéru. Ďalším veľmi dôležitým aspektom vývoja riadeného testami je aj to, že takto vytvorený kód je logicky rozdelený do malých jednotiek, ktoré je možné samostatne testovať. Dalo by sa to prirovnať k princípom objektovo-orientovaného programovania, pretože kód je logicky rozdelený do určitých modulov, ktoré je možné znovupoužiť v budúcnosti. Čo sa týka samotného princípu

vývoja riadeného testami, je dôležité si uvedomiť, že kvalita a funkčnosť celého vytvoreného zdrojového kódu vlastne závisí od vytvorených testov a ich kvality. Nesprávne vytvorený test vedie k vytvoreniu nesprávneho zdrojového kódu.

Kvalita zdrojového kódu a refaktoring

Akú úlohu v manažmente kvality zohráva kvalita zdrojového kódu? Myslím, že táto otázka je v dnešnej dobe veľmi aktuálna a napriek tomu sa tejto problematike venuje len veľmi málo pozornosti.

Zjednodušene by sa dalo povedať, že v komerčných softvérových produktoch na kvalite zdrojového kódu takmer vôbec nezáleží. Keď sa na to pozrieme z pohľadu klasickej definície kvality, tak kvalita softvérového produktu sa hodnotí predovšetkým ako miera splnenia požiadaviek zákazníkov. No to, čo zákazník požaduje, je výsledná aplikácia a nie zdrojový kód a z pohľadu kvality je teda nepodstatné, aká je jeho kvalita. Takisto ani kompilátoru „nezáleží“ na kvalite zdrojového kódu a vie vytvoriť ekvivalentný binárny súbor z „pekného“ aj „škaredého“ zdrojového kódu.

Existujú určité základné pravidlá ako písať kvalitný zdrojový kód. Sú to určité princípy, ktoré je vhodné dodržiavať. Napriek tomu, že pre výsledný produkt je kvalita zdrojového kódu nepodstatná, je veľmi dôležitá pre vývojárov, ktorí budú na projekte robiť v budúcnosti. Myslím, že najmä manažéri softvérových projektov by si mali uvedomiť, že kvalita zdrojového kódu nepriamo vplýva aj na kvalitu celého systému a mali by trvať na tom, aby programátori dodržiavali aspoň základné pravidlá štruktúrovania a písania zdrojového kódu. Medzi takéto pravidlá môžeme zaradiť napríklad dodržiavanie určitých konvencií pri pomenovaní premenných v programe a vhodné komentovanie programu. Samozrejme tých pravidiel nemôže byť príliš veľa, nesmú byť zložité, aby nepôsobili na programátorov obmedzujúco. Toto by potom mohlo do istej miery narušiť tvorivý proces, ktorým programovanie určite je.

Jeden z možných spôsobov ako udržiavať zdrojový kód kvalitný, je refaktoring. Je to technika na vylepšenie kódu bez zmeny jeho správania. Jeho hlavným cieľom je upraviť zdrojový kód tak, aby mal lepšiu štruktúru, bol prehľadnejší ale zároveň nenarušiť jeho funkcionalitu. Takto upravený zdrojový kód väčšinou pozostáva z menších a ucelených logických celkov a tieto sú potom vhodné pre znovupoužitie.

Záver

Problematika kvality softvérových systémov je veľmi rozsiahla a je pomerne zložité určiť, ktoré metódy manažmentu kvality sú vhodné pre konkrétne softvérové projekty.

V tejto práci som sa snažila popísať práve tie aspekty kvality softvéru, ktoré sú dôležité pri riadení softvérového projektu podobného tomu, ktorý riešime na našej fakulte. Sú to práve agilné metodiky vývoja, ktoré považujem za zaujímavé. Myslím si, že riadenie sa niektorými ich princípmi môže byť veľmi nápomocné pri riešení nášho projektu. Asi najdôležitejší atribút kvality v takomto projekte je práve kvalita zdrojového kódu. Je to z toho dôvodu, že vývoj v tejto oblasti umelej inteligencie stále

napreduje a projekt, ktorý budeme vyvíjať, bude mať „životnosť“ pravdepodobne niekoľko rokov a zakaždým bude s tým istým zdrojovým kódom pracovať iný tím študentov. Ak by sa nedodržiavali aspoň základné princípy písania kvalitného zdrojového kódu, po čase by tento kód bol taký neprehľadný, že by sa stal takmer nepoužiteľným a celý vývoj by sa musel robiť od začiatku.

Samozrejme okrem kvality zdrojového kódu sú tu aj iné aspekty kvality, na ktoré treba dbať a to napríklad kvalita dokumentácie. To už však môže byť námetom pre ďalšiu prácu.

Použitá literatúra

1. Bieliková, M.: *Softvérové inžinierstvo. Princípy a manažment*. Vydavateľstvo STU, Bratislava 2000.
2. Evans,I..: *Achieving Software Quality through teamwork*. Artech House, 2004.
3. Mugridge, R.: *Test Driven developlment and the Scientific method.* New Zaeland http://agile.openmex.com/agile2007/2003/files/P6Paper.pdf

Annotation

How to find a suitable quality assurance system for the specific software project

In the contemporary age there’s increase of using of informatics in our society. Using of software spread into all areas of our life. Lot of companies is becoming highly dependent on using of software products. For that reason the importance of software system quality is growing. The quality of software systems is a very discussed subject of many conversations and there’s a lots of procedures, models and methodologies for quality assurance management. Software projects are very different and each of them has got a different quality requirements. For that reason it’s very important to choose an appropriate system for quality management which is dependent on the nature of each project, on the … of team and other attributes. In this essay I will focus on quality assurance in small software projects which are similar to projects developed on our faculty.