Budeme mať kvalitný softvér?

DUŠAN RODINA

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

Abstrakt. Spôsob ako zabezpečiť absolútnu bezchybnosť softvérových produktov v dnešnej dobe nepoznáme. Existuje však množstvo metód na zvýšenie kvality produkovaného softvéru. Napriek tomu nie sú systematicky využívané, hlavne v malých firmách. Dôsledkom sú problémy s testovaním a nedostatočná kvalita produktov prejavujúca sa hlavne prítomnosťou väčšieho počtu chýb. Preto je veľkou výzvou celej informatickej spoločnosti, nielen vymýšľanie nových lepších metód testovania a zlepšovania kvality, ale aj hľadanie spôsobov ako ich preniesť do praxe. Cesta k všeobecne kvalitnejšiemu softvéru by mohla viesť cez skvalitňovanie nástrojov používaných pri jeho vývoji, pretože to, čo dáva ľuďom silu v najrôznejších oblastiach, sú hlavne efektívne nástroje znásobujúce schopnosti jednotlivých osôb.

Úvod

Otázka, či je možné zaručiť kvalitu v softvérových projektoch a hlavne v ich výstupoch – v softvérových produktoch, trápi odborníkov v oblasti informatiky už niekoľko desaťročí. Odpoveď nemusí byť tak blízko, ako by sme chceli. V odvetviach ako architektúra, stavebníctvo, či potravinárstvo sú jasne definované normy, ktoré umožňujú zabezpečiť požadovanú kvalitu výstupov. Tieto odvetvia sú však aj oveľa staršie ako informatika a mali nie desaťročia ale storočia na svoj vývoj. Pri softvéri navyše dosahujeme zložitosti a komplexnosti, ktorá rádovo prevyšuje zložitosť produktov v iných oblastiach.

Základným rámcom, ktorý umožňuje softvérovým inžinierom a vývojárom definovať kvalitu softvéru je norma ISO 9126. Tá definuje šesť vlastností (podrobnosti v tabuľke 1), ktoré by mal spĺňať kvalitný softvérový produkt – funkčnosť, spoľahlivosť, použiteľnosť, efektivita, udržovateľnosť a prenositeľnosť. Vieme ich však aj dosiahnuť? V absolútnej miere alebo len čiastočne? Dokážeme vôbec na základe nejakých meraní odlíšiť kvalitný softvérový produkt od toho nekvalitného?

Odpovede na tieto otázky sa snažia nájsť odborníci po celom svete prostredníctvom rôznych výskumných projektov a prác. Výsledky potom vyúsťujú do

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-8.

nových metód a techník, ktoré zlepšujú kvalitu procesu vývoja softvérových produktov a nepriamo aj samotný softvér.

Tab. 1. Vlastnosti a čiastkové vlastnosti kvality definované normou ISO/IEC 9126

Vlastnosť	*Čiastkové vlastnosti*
Funkcionalita	Vhodnosť, presnosť, interoperabilita, bezpečnosť.
Spoľahlivosť	Vyzretosť, tolerancia k chybám, obnoviteľnosť.
Použiteľnosť	Porozumiteľnosť, naučiteľnosť, funkčnosť, atraktívnosť.
Efektivita	Časová efektivita, využitie zdrojov, naplnenie efektivity.
Udržovateľnosť	Analyzovateľnosť, meniteľnosť, stabilita, testovateľnosť.
Prenositeľnosť	Prispôsobivosť, inštalovateľnosť, nahraditeľnosť, spolužitie.

Rôzne prístupy, rovnaký cieľ

Najväčšou prekážkou pri zlepšovaní kvalitatívnych parametrov softvéru je jeho komplexnosť a zložitosť. Je známe, že ľudský mozog dokáže v jednom momente pracovať so 7 ± 2 jednoduchými objektmi. Dokonca pokiaľ ide o abstraktné pojmy znižuje sa tento počet len na 2 – 3 objekty [4], čo sa javí byť dosť veľké obmedzenie pri tvorbe zložitých systémov ako sú tie softvérové.

Našťastie sme nerezignovali a existuje množstvo rozmanitých prístupov, ktoré sa snažia korigovať nedostatky ľudí ako jednotlivcov, v dôsledku čoho môžeme dosiahnutie zlepšenie kvality. Niektorým prístupom sa to darí viac, niektorým menej, dodnes však neexistuje taký, ktorý by vedel zaručiť absolútnu kvalitu. Najlepšie je teda všetky prístupy vhodne kombinovať.

Testovanie

Najčastejšie používaný a mnohokrát veľmi preceňovaný prístup je testovanie. Dokáže v najrôznejších oblastiach zabezpečiť, aby sa chybné produkty nedostali k zákazníkovi. So zložitosťou softvéru si však celkom nevie rady. Parametre produktu ako efektivita a spoľahlivosť sa po jeho aplikovaní zlepšia, nie však na 100% úroveň. Navyše vlastnosti vnútornej kvality ako udržovateľnosť, čitateľnosť kódu nezlepšuje vôbec. Preto si myslím, že by nebolo rozumné spoliehať sa na testovanie ako na akýsi všeliek, ktorý nás ochráni pred chybami. V praxi som sa však aj ja podobne ako Les Hatton [2] stretol s výrokom: „Náš softvér neobsahuje chyby – otestovali sme ho.“ Toto konštatovanie býva pri najbližšej možnej príležitosti vyvrátené.

Z rozhovorov s kolegami ako aj z vlastných skúseností viem o viacerých nevhodných prístupoch pri nasadzovaní testovania. Priebežné testovanie zvykne bývať často podceňované, následkom čoho sa produkt testuje v neskorých fázach vývoja — zvyčajne keď sa fáza implementácie blíži ku koncu. Taktiež situácie, keď programátori nahrádzajú aj testerov, nie sú vôbec neobvyklé. Príčinu vidím v prílišnej snahe šetriť

(hlavne v malých firmách a vo firmách, kde informatické oddelenia predstavujú len časť celej spoločnosti), pričom sa cena tlačí nadol na úkor kvality produktu.

Spomenuté problémy, môžu byť riešené uplatnením testami riadeného vývoja, kedy je zabezpečené permanentné automatizované testovanie softvéru. Jeho výhodou je, že núti programátorov opravovať chyby už v skorých fázach vývoja. Spomenutá metóda by mohla byť ešte vylepšená, keby testy a program písali odlišné osoby. Zaviedla by sa akási súťaž, keď by sa programátor programu viac snažil predchádzať chybám a programátor testov napísať lepšie testy, ktoré by odhalili chyby ukryté v programe. Navyše programátor testov by nebol ovplyvnený myšlienkami, ktoré boli využité pri tvorbe programu.

Matematický dôkaz ako záruka kvality

Pri výskumných projektoch (hľadanie liekov, fyzikálne simulácie, RoboCup) môžeme algoritmy skontrolovať použitím matematickému dôkazu. Táto metóda je zo všetkých asi najnákladnejšia. Potrebujeme totiž odborníkov, ktorí budú vedieť dôkaz vypracovať. Navyše môžeme túto metódu použiť iba na najkritickejšie časti programu, keďže rozsiahle časti kódu sú príliš komplikované. Navyše sa doba vývoja softvéru pri dôkaze predlžuje o nezanedbateľný počet hodín. Preto si myslím, že testovanie predstavuje vhodnejšiu metódu na zaručenie kvality, pokiaľ nejde o kritické oblasti, v ktorých chyba môže mať fatálne následky.

Kvalitná databáza ako základ

Michael Blaha vidí cestu ku kvalitnejšiemu softvéru hlavne v zlepšovaní dátového modelu a databázy všeobecne [1]. Zlepšovanie týmto smerom pokladám za dobrú myšlienku. Kvalitne navrhnutá databáza umožňuje sústrediť sa pri vývoji softvéru na podstatu veci. Naopak chybný návrh vedie k produkovaniu nadbytočného kódu, ktorého úlohou je hlavne plátať nedostatky databázového modelu.

V praxi som sa stretol s prípadom nie veľmi kvalitného návrhu databázy, ktorý sa oblúkom vyhol takmer všetkým štandardom – nemal definované žiadne identity, primárne ani cudzie kľúče. Pomerne veľa názvov bolo v tvare, ktorý nedával veľmi tušiť, čo sa za nimi skrýva. Pomenovania boli vo viacerých prípadoch nekonzistentné. Vymenované nedostatky spôsobovali komplikácie počas vývoja a značne ho predlžovali.

Naopak pri projekte, kde sa dátovému modelu venovala veľká pozornosť, som jasne pozoroval menší výskyt chýb aj v programe. Riadne definované vzťahy medzi jednotlivými entitami zabránili chybám, ktoré by mohli mať ďalekosiahle negatívne dôsledky. Konzistentné pomenovania zároveň zjednodušili prácu s databázou a dobu potrebnú na naučenie sa jej štruktúry.

Aj keď sú typy projektov, pri ktorých databáza nezohráva žiadnu úlohu – napríklad rôzne projekty umelej inteligencie, výskumné projekty, hry – existuje podstatne väčšia množina takých projektov, kde predstavuje databáza jadro celého vyvíjaného systému. Kvalitný dátový model je preto jednou z najdôležitejších vlastností, ktoré musí kvalitný softvér spĺňať.

Dokumentácia

Písanie dokumentácie nepatrí k obľúbeným činnostiam programátorov, preto býva veľmi zanedbávané. Jeho absolútne ignorovanie môže mať dosť závažné dôsledky. Bol som neraz svedkom toho, ako absolútna absencia materiálov viedla k závislosti na konkrétnych ľuďoch, ktorí pôsobili v danej firme a s tým spojené komplikácie po ich odchode.

V ďalšom prípade problémy vznikali pri zaúčaní nových členov tímu, ktorí nemali k dispozícii žiadne materiály a všetky podrobnosti im museli byť vysvetľované ústnym podaním. Rovnaká situácia sa opakovala pri odovzdaní produktu testerom. Chýbajúca technická dokumentácia tak spôsobila predĺženie a následné predraženie vývoja.

Existujú metódy, ktoré spájajú písanie dokumentácie a programu do jedného celku. Známa, no v praxi nepoužívaná metóda je literate programming. Umožňuje vytvárať programy, ktoré sú ľahko pochopiteľné a teda aj udržovateľné. V reálnych projektoch sa však používajú metódy, ktoré pristupujú k vývoju tradičným spôsobom, pričom do kódu sú vkladané komentáre v špeciálnom tvare, umožňujúce automatické vygenerovanie technickej dokumentácie. Tie sa ukázali ako životaschopnejšie ako literate programming v čistej forme.

Refactoring

Refactoring považujem za prelomovú metódu pri vývoji softvéru. Vďaka nástrojom s jeho podporou sme schopní vytvárať softvér s kvalitnejšou vnútornou štruktúrou. Premenovávanie metód a premenných spolu s ďalšími možnosťami reorganizácie zdrojového kódu vedú k lepšej udržovateľnosti softvéru, k znižovaniu zložitosti a následne k menšiemu výskytu závažných chýb.

Údržbu vlastného zdrojového kódu som vykonával aj predtým, ako som vôbec slovo refactoring prvýkrát počul. Bola však veľa prácna, zaberala veľa času a bývala pôvodcom nových chýb v programe. To ma (a určite aj veľa iných programátorov) odrádzalo od takejto činnosti. Vždy som si radšej dvakrát rozmyslel, či sa do daného prerábania kódu pustím. Refactoring k podobným úvahám nevedie a vďaka nemu zdrojový kód doslova žije a vyvíja sa, nielen nekontrolovane rastie.

Agilný vývoj softvéru

Pojem agilný vývoj v sebe zahŕňa viacero metód a techník. Hlavným cieľom[1] je zabezpečiť spokojnosť zákazníka pravidelným zasielaním nových verzií softvéru. Významnou črtou agilného vývoja, ktorá sa často využíva v praxi, je tiež zvýšený dôraz na priamu komunikáciu v tíme a hlavne s klientom.

Agilný vývoj neprináša v zásade žiadne nové prevratné myšlienky, je zbierkou viacerých metód a techník, ktoré sa postupne a prirodzene vyvinuli v praxi. Aj napriek tomu si myslím, že agilný spôsob vývoja softvéru predstavuje veľmi pozitívny prínos

[1] http://www.agilemanifesto.org/principles.html

k skvalitneniu softvéru. Vytvára totiž komplexný celok metód, ktorých sila spočíva hlavne v ich vzájomnej kooperácií.

Vplyv ľudského faktora

Pri všetkých metódach, ktoré som spomenul zohráva ľudský faktor dôležitú úlohu. Nepoznáme zatiaľ žiadny spôsob ako tento vplyv zásadne zmierniť. Rádové rozdiely vo výkonnosti jednotlivcov majú za následok slabé možnosti predvídania výsledku softvérového projektu. Šestnásťročný programátor bez akademického vzdelania môže dosahovať lepšie výsledky ako jeho kolega s doktorátom [2]. Z tohto pohľadu je tvorba softvéru viac umenie ako inžinierstvo.

Pokiaľ sa pri softvérových projektoch budeme spoliehať výhradne na jednotlivcov, zaručiť kvalitu nikdy nebudeme vedieť. Z vlastnej skúsenosti môžem potvrdiť, že nielen medzi schopnosťami dvoch ľudí sú veľké rozdiely, ale aj schopnosti danej osoby sa môžu v čase odlišovať. Nemyslím teraz len na rozdiely v rámci jedného dňa, ale aj v dlhších časových obdobiach. Syndróm „vyhorenosti“ môže byť dôsledkom zníženia výkonnosti ľudí, na ktorých sa v svojich plánoch spoliehame.

Prostredie, v ktorom ľudia pracujú takisto vplýva na ich výkonnosť. Príliš veľa rušivých momentov vedie k nesústredenosti a tým pádom k produkovania menej kvalitného softvéru. Ak chceme využiť ľudský faktor na maximum, musíme pre členov softvérového tímu zabezpečiť vhodné podmienky na prácu.

Kvalitný výsledok s kvalitnými nástrojmi

Keď dáme ľuďom do rúk dokonalejšie nástroje, rozdiely medzi nimi sa začnú postupne zmazávať. Vďaka kvalitnejších nástrojom na vývoj softvéru môžeme dnes dosahovať lepšie výsledky ako pred niekoľkými rokmi. Aj keď posun kvality nie je veľmi výrazný, je určite prínosom. Michael Blaha vo svojom výskume zistil, že v oblasti databázových technológií vzrástla kvalita (vyhodnotená na základe oznámkovania určených vlastností) o 30 % v období 1997 – 2002 oproti obdobiu 1992 – 1997, čo pripisuje kvalitnejším nástrojom na vývoj [1].

Moderné vývojárske nástroje naznačili pozitívny trend v oblasti zlepšovania kvality. Dnešné kompilátory dokážu automaticky označiť množstvo potenciálnych chýb, ktoré by inak unikli pozornosti programátorov. Prípady ako nepoužité premenné, neinicializované premenné, nedostupný kód, funkcie s vetvami bez definovaného výstupu a mnohé ďalšie sú dnes potlačené na minimum.

Osobne by som uvítal nástroje, ktoré by automaticky sledovali, ktoré moduly môžu byť potenciálne ovplyvnené uskutočnenými zmenami v kóde. Na prvých priečkach v zozname by boli tie moduly, ktoré sú zmenami ovplyvnené v najväčšej miere. Označené by boli miesta, kde môže teoreticky dôjsť k problémom a týmto by som ako programátor mohol venovať viac pozornosti.

Predpokladám, že automatické inteligentné odhaľovanie chýb a potenciálnych problémov v programe sa bude ďalej vyvíjať a tak umožní zlepšiť kvalitu výstupov

slabších programátorov, či odstrániť prehliadnuté nedostatky v kóde skúsených vývojárov.

Určenie kvality

Ak nemáme dostatočne kvalitné nástroje, metódy a techniky na dosiahnutie požadovanej kvality softvéru, musíme sa snažiť aspoň predpovedať, akú úroveň kvality sme schopní dosiahnuť v danom projekte s danými zdrojmi a o koľko ju môžeme zlepšiť rôznymi dostupnými prostriedkami. Táto vedomosť by nám umožnila dosiahnuť pri projektoch dobrý pomer cena/výkon.

Pri meraní kvality softvéru sa zohľadňuje hlavne kvalita zdrojového kódu a dokumentácie. Dôležitým parametrom je počet riadkov zdrojového kódu, keďže je to jeden z mála objektívnych parametrov. Ako jedna sumárna hodnota opisujúca celkovú veľkosť programu nemá veľký zmysel. Pokiaľ sa však používa v pomeroch s inými parametrami (počet tried, počet metód, dĺžka riadku), dostávame relevantnejšie výsledky, ktoré nám môžu poodhaliť dizajnové nedostatky softvéru.

Okomentovanosť kódu dokáže významne zlepšiť čitateľnosť kódu a jeho udržovateľnosť. Automaticky merať tento parameter nie je také jednoduché, ako si to myslia viacerí autori rôznych analytických nástrojov. Okomentované časti kódu totiž vôbec nemusia popisovať nasledujúci kód, ale môže predstavovať zakomentovaný nepoužitý kód [3] a v štatistikách sú veľmi zavádzajúce.

Súbor viacerých parametrov (previazanosť modulov, priemerná veľkosť metód, počet metód v triedach, úroveň vnorenia, …), ktoré vieme pri softvéri merať, nám umožní zistiť jeho približnú kvalitu. Zaznamenávaním histórie týchto parametrov by sme mohli odhadnúť aj budúci kvalitatívny vývoj produktu a podľa toho riadiť zdroje, ktoré máme k dispozícii.

Uvedenie do praxe

Uviesť do praxe nové metódy, techniky, či nástroje, nie je vôbec také jednoduché, ako si môžeme spočiatku myslieť. Firmy sa často noviniek obávajú, keďže nechcú riskovať, že sa vydajú cestou, ktorá síce vyzerá sľubne, ale časom sa ukáže, že vedie do slepej uličky a neprináša im žiadne benefity. Preto vyčkávajú, ako sa situácia vyvinie a dovtedy používajú overené prostriedky.

Za značný problém pri zavádzaní noviniek do praxe pokladám nedostatok odborníkov, ktorí vedia využívať nové prístupy, či pracovať s novšími nástrojmi a naplno využiť ich potenciál. Bežné sú prípady, keď firma zabezpečí moderný kvalitný softvér, zamestnanci však nevyužívajú jeho nové možnosti a teda stagnujú na rovnakej úrovni kvality. Aj preto sa dnes dobré univerzity nesnažia študentov informatiky naučiť pracovať s konkrétnymi nástrojmi, ale skôr vybudovať u nich schopnosti samostatne sa učiť stále nové veci.

Na výber je dnes veľké množstvo metód na zlepšenie kvality. Ideálne by bolo keby sme použili všetky, ktoré sa osvedčili. Skutočný stav má od toho ideálneho

ďaleko. V reálnom svete sa totiž musí rátať s finančnými a časovými obmedzeniami. Honba za splnením termínu má často za následok zhoršenú kvalitu produktu.

Ako teda zabezpečiť produkovanie softvéru vysokej kvality v takýchto sťažených podmienkach? Riešenie vidím v udeľovaní certifikátov kvality a ich popularizácii tak, aby sa dostali do všeobecného povedomia ľudí. Zákazník by si potom vybral takú firmu, o ktorej by vedel, že pri vývoji softvéru dokáže zaručiť určitú úroveň kvality. Firmy by boli motivované uplatňovať viac metódy ako agilný vývoj, extrémne programovanie, testami riadený vývoj, používať moderné inteligentnejšie nástroje a tiež by viac investovali do vzdelávania svojich zamestnancov. Takéto zmeny by síce vyžadovali značné finančné prostriedky, v konečnom dôsledku by ale priniesli úspory – kvalitnejší softvér by nevyžadoval toľko dodatočných zásahov potrebných na opravy chýb.

Budeme mať teda kvalitný softvér?

Doterajší vývoj smeruje pomalým tempom k softvéru vyššej kvality. Je možné, že nikdy nedosiahneme absolútnu kvalitu, ale myslím si, že sa k nej časom priblížime natoľko, že bude väčšina ľudí presvedčená o jej dosiahnutí.

Stačí si uvedomiť, ako ďaleko sme pokročili za krátky čas existencie informatiky. Na začiatku boli programy, ktoré sa ovládali nepohodlne a správali sa nespoľahlivo pri každej reakcii používateľa, ktorá nebola vykonaná presne podľa manuálu. Teraz máme k dispozícií používateľský prívetivé aplikácie, na ktorých spoľahlivosť sa už nemôžeme tak často sťažovať. Je to práve dôsledok stále častejšieho využívania nových metód ako aj moderných softvérových nástrojov.

Na záver by bolo vhodné dať odpoveď na otázku nastolenú v tejto práci. Myslím si, že softvér budeme mať kvalitný do takej miery, do akej budeme mať kvalitné nástroje na jeho vývoj. Preto verím, že vďaka rýchlej evolúcii v tejto oblasti, bude odpoveď jedného dňa znieť bez akéhokoľvek zaváhania – áno.

Použitá literatúra

1. BLAHA, Michael: A Copper Bullet for Software. In: *Computer*, 37/2004, s. 21 - 25.
2. HATTON, Les: The Chimera of Software Quality. In: *Computer*, 40/2007, s. 102 - 103.
3. HEITLAGER, I., KUIPERS, T., VISSER, J.: A Practical Model for Measuring Maintainability. *Quality of Information and Communications Technology, International Conference*. 6th International Conference on the Quality of Information and Communications Technology, 2007, s. 30 - 39.
4. ROSSER Peter. Thoughts on software complexity. [online] Publikované 2.6.2006. [citované 16.10.2008]. Dostupné z http://blogs.msdn.com/peterrosser/archive/2006/06/02/SoftwareComplexity.aspx

Annotation

Will we have quality software?

Nowadays, we do not know the way, how to assure absolute perfection of software products. However, there is big amount of methods, which improves quality of produced software. In spite of it, these methods are not utilized, especially in small companies. The consequences are complications with testing and the insufficient product quality, when products contain lot of errors. That is why the inventing of new better methods of testing and improving of quality is big challenge for whole computer society, as well as the finding of way how to bring this progress in the use. We can develop more quality software using more quality tools, because the tools are exactly the things, which give the power to humans in different areas and improve their abilities.