KEĎ CHYBA PREDSTAVUJE POZITÍVUM

Dokonalosť neexistuje. Existuje akceptácia chyby a poučenie sa z nej.

Ondrej Grman

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
o.grman@gmail.com

Abstrakt. *Softvér je komplexné dielo, s veľkým priestorom, kde sa môžu vyskytnúť chyby. Práca pojednáva o rôznych spôsoboch, ako tento priestor eliminovať, ako sledovať proces chyby, hľadať kritické miesta pre vznik chýb, odhaľovať ich ešte v zárodku. V práci sledujem rozdiely medzi výskytom a detekciou „veľkých" chýb, teda tých, ktoré sú viditeľné jednak z pohľadu ich odhaľovania , ale viditeľné aj v podobe následkov, ktoré ich zanedbanie prináša. Ďalej sa práca sústreďuje na fenomén chýb z nedbanlivosti, nepozornosti, pri ktorom vzniká drvivá väčšina nedostatkov softvéru, pri prvom pohľade zdanlivo nepodstatných, no v konečnom dôsledku pri nevhodne kombinácií a veľkom množstve pre správnosť fungovania softvéru chýb kritických.*

Kľúčové slová: *chyba, softvérové metriky, kvalita, znovupoužiteľnosť, udržateľnosť*

Úvod

Sledovanie procesu výroby akéhokoľvek výrobku býva často v niektorých aspektoch zložitejšie ako samotný výrobok. Ak pri výrobku vieme odmerať jeho rozmery, opísať jeho zloženie alebo ďalšie vlastnosti, opísať proces – nehmotnú vec, treba nemálo úsilia. Toto platí v plnej miere aj pre stanovovanie procesov riadenia kvality pri vývoji softvérového riešenia. Pri softvéri je potrebné sústrediť sa na množstvo častí, z ktorých sa celok skladá, tieto časti vzájomne správne poskladať a zabezpečiť, aby ako jeden celok prinášali pre svojho budúceho zákazníka hodnoty, ktoré od produktu očakáva. Softvérový produkt je

čo do výroby náročnejší, nie je potrebné iba zabezpečiť jednotlivé časti, poskladať ich spolu, problémom softvérových produktov je ich abstraktnosť. My ľudia sme zvyknutí veci uchopovať a je nám akosi prirodzené rozhodovať sa o kvalite, nekvalite na základe našich zmyslov- vnemov, dotyku, farbe, vôni.

Ako ho potom otestovať, ako určiť, aký kvalitný je? Po desaťročiach vytvárania rôznych systémov a softvérov sa dá povedať, že kvalita je veľmi podstatný aspekt a tiež metrika každého systému. K plnému pochopeniu kvality ako pojmu a tiež kvality ako metriky projektu je potrebne pochopiť prečo je taký pojem vlastne taký potrebný.

Kvalita softvéru, čo to je ?

Kvalita softvéru je hlavne miera splnenia potrieb zákazníka. Požiadavky na softvér sú rôzne. Jednak ide o požiadavky zákazníka, ktorý od produktu očakáva, že bude robiť to, čo od neho chce, jednak ide aj o požiadavky vývojárov, kam patrí najmä ľahká udržateľnosť a znovupoužiteľnosť. Takýchto očakávaní je samozrejme viac, uvádzam iba tie najrelevantnejšie. V praxi to znamená, že softvér je pod drobnohľadom jednak vývojárov a jednak zákazníkov, pričom obaja sledujú svoje záujmy. Tie nie sú vždy totožné a preto každý softvér spĺňa iba určitý pomer medzi splnením očakávaní zákazníka a vývojára.

Norma ISO 9000:2000 definuje kvalitu ako súhrn vlastností a charakteristík výrobku, procesu služby, ktoré preukazujú jeho schopnosť plniť určené potreby. Tento dokument určuje aj sústavu noriem pre riadenie a zabezpečovanie kvality, ktoré sú medzinárodne dané a je možné ich posúdiť aj z externého prostredia [5].

Kvalita v sebe skrýva široké spektrum možností ako ju posudzovať, pre jedného môže byť atribútom kvality iná vlastnosť, akú za kvalitatívny indikátor považujú druhí.

V oblasti informačných technológií a špeciálne pri vývoji softvérových produktov bolo a je pre ďalší vývoj a posun v oblasti dôležité stanoviť jednoznačné parametre identifikujúce kvalitu.

Ak posudzujeme kvalitu softvéru, nemali by sme skĺznuť iba do roviny testovania správnosti, funkčnosti programu iba ako zhluku zdrojového kódu predstavujúceho zapísané myšlienky programátorov.

Prečo robíme chyby, keď vieme, ako ich nerobiť

Dôležitým je aj pohľad pod kapotu softvérového tímu, nemálo krát sú chyby softvéru spôsobené nie tým, že samotný kód jednotlivých programátorov podieľajúcich sa na vývoji riešenia ako celku je chybný, ale tým, že programátori síce všetci ako jednotlivci dodržali všetky štandardný a normy, no ako kolektív ich nepretransformovali do celku.

Stáva sa totiž, že pri zlom interpretovaní a výmene informácií v tíme sú zadefinované postupy a všeobecná konvencia, dokonca aj myšlienkový prúd tímu je nastavený rovnako, no dôjde k drobnej odchýlke zo strany niektorého z členov tímu a môže nastať problém.

Samozrejme, takéto situácie by mali byť odhaľované takmer okamžite pri procese testovania, a to v rôznych štádiách od jednotkového testovania pri programovaní, neskôr v závislosti na metodike vývoja pri testovaní v rôznych fázach rozpracovanosti riešenia. No chyby sa dejú stále.

Mýliť sa je ľudské

My ľudia sme kreatívni, schopní prinášať nové myšlienky, postupy, na druhej strane sú našou prirodzenou vlastnosťou, keď aj nie priamo nezodpovednosť, ale prinajmenšom náchylnosť vytvárať chyby z nepozornosti a v takýchto prípadoch dochádza k situáciám, kedy hoci aj vývoj softvéru je správne riadený, riziká sú eliminované, procesy sú správne riadené, kvalita je zabezpečovaná a strážená, no ľudský faktor zlyhá, vytvorí problémovú situáciu.

Ako eliminovať zlyhanie človeka?

V praxi, pri vývoji programových riešení, tak ako aj v iných odvetviach sú firmy nútené kvôli snahe získať zákazky, zabezpečiť prácu pre svojich ľudí a v neposlednom rade generovať zisk, uchýliť sa k aplikovaniu metód, ktoré nie sú vhodné. Časové plány na vývoj jednotlivých riešení sú skresávané na minimum, tímy bývajú poddimenzované, trpia nedostatkom technického vybavenia.

Niet sa čo čudovať, že v takomto prostredí dochádza k zlyhávaniu pracovníkov, nie v závažných veciach, na ktoré sú v dobrých spoločnostiach vytvorené mechanizmy zabraňujúce vzniku kritických chýb, ale vo veciach, drobných, často zdanlivo triviálnych a nepodstatných. Tu ale klameme sami seba. Ak dochádza k hromadeniu drobných chybičiek označovaných ako nepodstatné, v celkovom meradle môžu zapríčiniť problém mnohokrát s následkami horšími ako so sebou nesú problémy veľké, kritické, ktoré eliminujeme, a na ktoré máme vybudované a v praxi otestované riešenia ako ich odfiltrovávať.

Dvakrát meraj, raz rež

Kto nič nerobí, nič nepokazí. Ako ale odmerať chybu ? Vieme si predstaviť, ako softvér otestovať, ale ako vieme odmerať jeho nechybovosť či kvalitu ?

Postupným a búrlivým vývojom v IT oblasti sa paralelne s vývojom techník ako písať program od istého okamihu objavujú aj spôsoby pomocou ktorých je možné kvalitu vyčísliť. Existujú však rôzne metriky, ktoré sú vhodné pri rozličnom pohľade na situáciu.

Aký meter vytiahnuť

Každý softvér vyvíjaný podľa zákazníckych požiadaviek je svojím spôsobom jedinečný a vyžaduje individuálny prístup k vývoju. Preto aj pri posudzovaní, akú metriku identifikácie kvality softvéru použiť pri tom-ktorom projekte, je nevyhnutné zvažovať, ktorý spôsob je v danej situácií najvhodnejší. Každá metrika je špecifická a prináša pre riadenie kvality projektu iný význam.

Ak hovoríme o kvalite programu, máme predovšetkým na mysli eliminovanie chýb, ktoré by softvéru uberali na jeho použiteľnosti. Pri sústredení sa na chyby vieme na základe štatistických odhadov z predošlého vývoja predpokladať, kde by chyby najčastejšie mohli vzniknúť – za podmienok, že vyvíjaný softvér je typologicky podobný s tými projektmi, ktorých štatistické informácie používame.

Využívanou metódou založenou na štatistike je metóda SSPI [5], ktorá sa na základe kategorizácie chýb podľa pôvodu, následne sa sčíta počet chýb podľa príslušného kategorického rozdelenia, stanoví sa cena za jednu chybu a v konečnej fáze sa analyzujú ceny chýb podľa kategórií a celý proces vývoja softvéru sa optimalizuje tak, aby sa eliminovala frekvencia výskytu chýb v tejto kategórii.

Na prvý pohľad sa takýto prístup zdá racionálny a nemožno mu racionalitu uprieť, no pri podrobnejšom skúmaní sa objaví niekoľko nedostatkov. Jedným z najväčších, ktorý predstavuje nevýhodu pre menšie či začínajúce softvérové domy je nedostatok štatistických dát, ktorý sa prejavuje v podhodnotenej alebo nadhodnotenej cene za chybu. Ako tento prístup optimalizovať aby fungoval správne aj za týchto podmienok? Riešenie by mohlo spočívať v systematickom zbere údajov, nielen v rámci jednej firmy, ale štatisticky, samozrejme v očistenej forme nevynášajúcej firemné tajomstvá v zozbieraných dátach podľa rovnakých projektov. Znie to príliš utopisticky? Že je to v dnešnej konkurenčne vyhrotenej dobe nereálne?

Nereálnosť je možné prekonať ekonomickou výhodnosťou, ktorú by takéto riešenie prinieslo. Kde je vôľa, tam je cesta – kde sú predpoklady pre ušetrenie značného množstva prostriedkov určených na znižovanie ceny za vývoj a opravu, tam aj manažéri tvrdo bojujúci proti konkurencií nachádzajú vôľu.

Ak hovoríme o kvalite softvéru, treba mať na zreteli dva aspekty ovplyvňujúce kvalitu. Tým najznámejším a aj prirodzene najjednoduchšie uchopiteľným je správnosť (korektnosť), čo predstavuje počet chýb na KLOC [2]. Táto metrika mala svoje opodstatnenie predovšetkým v minulosti, dnes, keď dĺžka kódu pri použití rôznych programovacích jazykov môže byť diametrálne odlišná, nemožno považovať softvér, ktorý má menší počet chýb na KLOC automaticky za kvalitnejší. Takýto prístup by bol v nadnesenom pohľade ďaleko nebezpečnejší, ako možno samotné chyby v programe.

Chyby sa vyskytujú všade, aj keď tím pri vývoji sústreďuje enormné množstvo úsilia na eliminovanie chýb, čo je predovšetkým pri vývoji softvérov s vysokým stupňom požadovanej bezpečnosti softvéru viac ako potrebné, softvér, ktorý obsahuje chyby zlyháva aj tu [6].

Nájsť chybu je jedna vec, ale kvalitný softvér by mal byť postavený tak, aby ju bolo možné za čo najmenšie náklady napraviť, ak zákazník zmení požiadavky upraviť.

Rôzne principiálne odlišné pohľady na hľadanie chyby sú fajn, no keď dôjde na lámanie chleba a treba nájsť ihlu v kope sena, nastávajú tie ťažšie problémy. Metriky na odhaľovanie chýb totiž treba použiť tak aby prinášali výsledky, nie iba preto aby sme si mohli zaznačiť že sme sa touto problematikou – kvalitou zaoberali.

Keď má meter rôznu dĺžku

Nie každý spôsob merania je vhodný na každý softvérový produkt. Na ten treba hľadieť ako na jedinečnú vec a tomu prispôsobiť aj výber metrík. Pokiaľ vytvárame nástroj sami pre seba a s určitosťou vieme povedať, že neexistuje hrozba v podobe vírusov či iných útokov a zároveň vieme ako produkt používať, nie je nutné zaoberať sa napríklad metrikou integrity.

Pri každom softvéri je však veľmi vhodné riešiť užitočnosť, čo podľa Gilba [4] je snaha zmerať používateľskú prívetivosť. Podľa môjho názoru je to vlastnosť, ktorá

definuje či daný produkt bude vôbec používaný, pretože neužitočný produkt, taký ktorý nič neuľahčuje, zákazníci nechcú.

Pri zákazníckych produktoch je dôležitá už aj vyššie spomínaná integrita, ktorú Gilb [4] definuje ako odolnosť voči náhodným alebo úmyselným vonkajším útokom. Je daná sumou súčinu pravdepodobnosti nenastania útoku a pravdepodobnosti neubránenia sa útoku systémom cez všetky typy možných útokov.

V neposlednom rade je podľa mňa dôležité zaoberať sa aj metrikou MTTC (mean-time-to-change), čo Gilb [4] definuje ako dobu potrebnú priebeh celého životného cyklu softvéru, a teda analýzu požadovaných zmien, návrh, implementáciou, testovanie a distribúciu medzi používateľov.

Záver

Softvér predstavuje komplikovanú nehmotnú znalosť viacerých ľudí pretavenú do reálneho riešenia. Práca v tíme a manažovanie kvality prináša so sebou potrebu sústredenia sa na dodržiavanie a predovšetkým správne nastavenie procesov tvorby softvéru. Existuje veľké množstvo rôznych metrík, noriem kvality, osvedčených prístupov ako by sa mal softvérový produkt tvoriť. V praxi vo firemnom prostredí je nasadené množstvo podporných prostriedkov na správu kvality, rovnako ako sú podrobne zadefinované procesy smerujúce k dodržiavaniu kvality. Výzvou smerovanou do radov manažérov kvality je predovšetkým dôsledné sústredenie sa na elimináciu drobných chýb, ktoré denne v práci robíme, a ktoré považujeme za drobnosti, ktoré nie sú hodné našej pozornosti. Toto je veľký omyl. Veľké chyby sú rýchlo odhaliteľné, my sa preto musíme sústreďovať na elimináciu malých, no často draho zaplatených chýb z nedbanlivosti, nesústredenia. Mnoho vývojárov má obavy priznať si chybu, no chyba, ktorá je odhalená predstavuje prínos. Ak si sami uvedomíme, kde robíme chyby, druhýkrát si dáme pozor.

Použitá literatúra

1. Kettunen, V., Kasurinen, J., Taipale, O., Smolander, K.: A study on agility and testing processes in software organizations. In *Proceedings of the 19th international symposium on Software testing and analysis* (ISSTA '10). ACM, New York, NY, USA, 231-240, 2010
2. Ogasawara, H., Yamada, A., Kojo, M.: Experiences of software quality management using metrics through the life-cycle. In *Proceedings of the 18th international conference on Software engineering* (ICSE '96). IEEE Computer Society, Washington, DC, USA, 179-188.
3. Nasib S. Gill. 2005. Factors affecting effective software quality management revisited. *SIGSOFT Softw. Eng. Notes* 30, 2 (March 2005)
4. Gilb, T. Principles of Software Project Mangement, Addison-Wesley, 1988
5. Norma ISO 9000
6. Erdbrink,T.: Iran Confirms Attack by Virus That Collects Information, New York Times, http://www.nytimes.com/2012/05/30/world/middleeast/iran-confirms-cyber-attack-by-new-virus-called-flame.html?_r=0

Annotation

When mistake is positive

Software is a complex piece with a lot of space where errors can occur. The work discusses the various ways to eliminate this space, to keep track of process errors, look for critical points of error, and exposing of them. I watch differences between the occurrence and detection of "large" errors in this paper, those that are visible both in terms of detection, but also visible in the form of consequences that their failure brings. The work focuses on the phenomenon of error of negligence, carelessness, which generates the vast majority of software defects. At first, glance seem insubstantial, but ultimately unsuitable for a large number of combinations for a correct functioning of critical software defects.