TESTOVANIE, CESTA K SPOĽAHLIVOSTI

Stopercentne spoľahlivý softvér je ako odvrátená strana Mesiaca. Každý vie, že existuje, ale videli ho asi iba v NASA.

Roman Bilevic

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

Abstrakt. *Spoľahlivosť je jedným z najdôležitejších parametrov kvality softvéru. Jej vyhodnocovanie je však často podceňované a odkladá sa až na neskoršie fázy vývoja. Taktiež treba brať do úvahy vplyv spôsobu vývoja softvéru na jeho spoľahlivosť. V tejto eseji opisujem hlavné úlohy manažovania spoľahlivosti. Ďalej analyzujem vývoj riadený testami (angl. test-driven development) z pohľadu spoľahlivosti a zamýšľam sa nad jeho výhodami oproti vývoju, kde testovanie prebieha až po ukončení implementácie. Opisujem, akým spôsobom sa tento typ vývoja vyrovnáva s chybami. Na základe predpokladov modelovania rastu spoľahlivosti softvéru uvažujem, či vývoj riadený testami tieto predpoklady spĺňa a do akej miery. Zhodnocujem, či je v tomto type vývoja vôbec potrebné modelovať spoľahlivosť vzhľadom na jeho povahu testovania.*

Kľúčové slová: *spoľahlivosť softvéru, model rastu spoľahlivosti, vývoj riadený testami, porucha, chyba*

Úvod

Jedným z dôležitých cieľov manažmentu kvality je dosiahnutie dostatočnej spoľahlivosti softvéru. Čo chápeme pod pojmom spoľahlivosť softvéru a ako vieme určiť, že je dostatočná? Definícia znie: „Spoľahlivosť softvéru je miera jeho schopnosti vykonávať bezporuchovú činnosť“[1]. Vyhodnocovanie spoľahlivosti softvéru počas jeho vývoja je však netriviálne. Spoľahlivosť softvéru sa zväčša určuje až po dodaní produktu

zákazníkovi a to na základe rôznych spätných väzieb. To je však podľa môjho názoru príliš neskoro, pretože na opravu príčin porúch musia byť v tejto fáze vynaložené omnoho vyššie náklady ako na ich opravu už počas vývoja softvéru.

Na vyhodnotenie spoľahlivosti softvéru ešte pred dodaním zákazníkovi preto slúžia rôzne modely spoľahlivosti softvéru. Tieto modely sa však môžu značne líšiť vo svojich výstupoch. Každý model má určité predpoklady, ktoré musia byť splnené, aby bol výsledok modelovania čo najpresnejší. Pri výbere modelu sa musí brať do úvahy povaha testovania, charakteristika softvéru, ľudský zásah a spôsob odstraňovania chýb.

Výber modelu spoľahlivosti a spoľahlivosť softvéru ako takú do veľkej miery ovplyvňuje aj spôsob vývoja softvéru. V dnešnej dobe je populárny vývoj riadený testami (angl. test-driven development), ktorý spočíva vo vytváraní testov ešte pred samotnou implementáciou. Vývojári sa následne pri implementácii sústredia iba na splnenie testov, čím sa odbúrava nadbytočná funkcionalita softvéru a zabezpečuje sa vysoké pokrytie testami. Ako takýto spôsob vývoja vplýva na spoľahlivosť softvéru a na výber modelu spoľahlivosti ? To sa pokúsim vysvetliť na nasledujúcich riadkoch.

Spoľahlivosť

Ako už bolo spomenuté v úvode, spoľahlivosť softvéru je vyjadrená pomerom času, kedy softvér vykonáva bezporuchovú činnosť a celkového času. Pre potreby vývoja softvéru sa však zväčša používa práve opačná miera, teda pomer času, kedy sa softvér nachádza v poruchovom stave a celkového času. Táto miera sa nazýva intenzita porúch alebo aj poruchovosť.

Na znižovanie intenzity porúch sú počas vývoja softvéru vynakladané dodatočné prostriedky umožňujúce odstrániť chyby softvéru, ktoré poruchy spôsobujú. Tu je potrebné si uvedomiť rozdiel v terminológii. Porucha (angl. failure) je odchýlka od toho, ako sa má softvér na základe požiadaviek počas prevádzky správať. Poruchy sú dynamické, pretože softvér musí vykonávať nejakú činnosť aby sa vyskytli. Chyba (angl. fault) je závada softvéru, ktorá spôsobuje poruchy [2]. Chybou môže byť napríklad zle napísaný kód programu.

Jednou zo základných úloh, ktorá musí byť vykonaná v rámci manažovania spoľahlivosti je určenie cieľovej intenzity porúch, ktorá predstavuje maximálnu poruchovosť tolerovanú zákazníkom a musí byť dosiahnutá v čase predania produktu zákazníkovi [3]. Ďalšími úlohami sú definovanie porúch produktu, odlíšenie softvérových porúch od tých hardvérových a určiť spôsob merania aktuálnej poruchovosti. Tieto úlohy sú do veľkej miery nezávislé od metódy vývoja softvéru. Závislý od spôsobu vývoja je však spôsob vyrovnávania sa s chybami. V zásade existujú tri spôsoby, ktoré sa v rôznej miere kombinujú. Je to prevencia chýb, odstraňovanie chýb a tolerovanie chýb, ktoré je však jasne obmedzené cieľovou poruchovosťou.

Vývoj riadený testami

Ak za normálny spôsob vývoja softvéru budeme považovať ten, ktorý zodpovedá vodopádovému modelu, tak odlišnosť vývoja riadeného testami je pomerne jasná. Zatiaľ čo v normálnom vývoji prebieha testovanie po implementačnej časti, tak pri vývoji

riadenom testami je implementácia a testovanie prepletené. Prvý sa vytvorí test , potom sa implementuje funkcionality softvéru a nakoniec musí táto implementácia prejsť už vytvoreným testom. Toto sa deje pre všetky časti implementovaného softvéru, takže na konci implementačnej fázy vývoja je už veľká väčšina funkcionality systému pokrytá testami. Aký to má však vplyv na spoľahlivosť?

Najväčší prínos z pohľadu spoľahlivosti vidím v spôsobe vyrovnávania sa s chybami. Pri normálnom spôsobe vývoja sa testuje až po implementácii, takže hlavný spôsob vyrovnávania sa s chybami je ich odstraňovanie. Pri testami riadenom vývoji sú testy vytvárané ešte pred implementáciou a vývojár sa pri implementácii sústredí iba na splnenie týchto testov. Takto sa zabraňuje implementácii zbytočnej funkcionality, ktorá býva najčastejším zdrojom chýb. Môžem preto povedať, že vývoj riadený testami do veľkej miery využíva prevenciu chýb a náklady na prevenciu chýb bývajú vo všeobecnosti nižšie ako náklady na ich odstránenie.

Za druhý najväčší prínos považujem jednoduchšie dosiahnutie cieľovej intenzity porúch. Keďže sa softvér implementuje po funkčných častiach a každá časť sa priebežne testuje, tak sa môže priebežne sledovať aj poruchovosť. Pokiaľ by bola poruchovosť vysoká, môžu sa prijať opatrenia ešte počas vývoja softvéru na jej zníženie. Tým pádom sa poruchovosť udržuje počas celého vývoja na prijateľných hodnotách. Pri normálnom vývoji by sa poruchovosť zisťovala až po testovaní a pokiaľ by nebola dosiahnutá cieľová poruchovosť, muselo by sa pristúpiť nie len k oprave chýb , ale často krát aj k pridaniu ďalších funkcií na zabezpečenie spoľahlivosti a tie by sa museli implementovať a otestovať, takže hrozí kaskádový efekt.

Povaha testovania má veľký vplyv aj na výber modelu rastu spoľahlivosti. Vzhľadom na špecifický prístup k testovaniu v tomto type vývoja je použitie modelu rastu spoľahlivosti otázne. Na základe predpokladov, ktoré majú modely rastu spoľahlivosti a charakteristiky vývoja riadeného testami, sa pokúsim určiť, či je vhodné modely rastu spoľahlivosti použiť.

Modely rastu spoľahlivosti

Väčšina modelov rastu spoľahlivosti má za úlohu vypočítať celkový počet chýb v určitej časti kódu [4]. Na tento výpočet používajú funkcie, ktoré zohľadňujú počet doposiaľ odhalených chýb a tempu objavovania týchto chýb za určitý čas. Dôležitý faktor pri rozhodnutí, či je softvér dostatočne spoľahlivý, aby bol dodaný zákazníkovi, poprípade koľko ďalších testov treba vykonať, aby sme dosiahli cieľovú poruchovosť je počet neznámych chýb. Tie sa dajú ľahko vypočítať rozdielom vypočítaného počtu celkových chýb a doposiaľ zistených chýb.

Modely rastu spoľahlivosti majú šesť základných predpokladov, ktoré by sa mali spĺňať, aby boli namodelované výsledky čo najpresnejšie. V praxi sa tieto predpoklady často porušujú, čo v konečnom dôsledku vedie k nesprávne odhadnutej spoľahlivosti softvéru. Tieto predpoklady sú [4]:

- chyby sa odstraňujú okamžite po ich nájdení,
- odstránenie chýb je úplné,
- chyby sú nahlasované iba jednou skupinou ľudí zodpovedných za testovanie,

- testy vychádzajú z prevádzkového profilu,
- počas testovania sa neprodukuje žiaden nový kód,
- testovacie úsilie sa nemení v čase.

Pri vývoji riadenom testami sa k oprave chýb pristupuje hneď po neúspešnom zbehnutí testu. Problém nastáva pokiaľ je vývoj riadený testami skombinovaný s niektorou agilnou metódou , ako napríklad SCRUM. Vtedy sa väčšie chyby označia ako nové úlohy a sú riešené až v ďalšej implementácii.

Cyklus implementácie a testovania jednej črty sa opakuje dovtedy, kým testy nezbehnú korektne, čo znamená, že chyby sú odstraňované úplne

To, že sú chyby nahlasované iba jednou skupinou ľudí zodpovedných za testovanie znamená, že sa do modelovania sa nezahŕňajú chyby nahlásené napríklad beta testermi alebo vývojármi pracujúcimi na ďalšom vydaní toho istého produktu. Pri vývoji riadenom testami sa nahlasujú chyby zistené pri testovaní jednotlivých implementácií a pri regresnom testovaní celej priebežnej implementácie. Oba druhy chýb sú rovnocenné a nahlasované testermi, takže môžem povedať, že tento predpoklad sa spĺňa.

Všeobecne sa predpokladá, že testovanie softvéru musí byť vykonávané podľa jeho prevádzkového profilu. To znamená, že testovacie vstupy sú vyberané podľa pravdepodobnosti ich výskytu počas prevádzky v danom prostredí. Tu nastáva prvý väčší rozpor, pretože vývoj riadený testami sa zameriava na funkcionálne testovanie. Jednotlivé funkcie by sa síce dali spájať do operácií vykonávaných zákazníkom a vďaka mock objektom by sa dala zaistiť ich nezávislosť na systéme, ale bolo by to náročné a oveľa menej efektívne ako prieme testovanie funkcionality.

Najzávažnejším rozporom vývoja riadeného testami a predpokladmi modelov rastu spoľahlivosti je pridávanie nového kódu počas testovania. Keďže je implementácia a testovanie prepletené, tak je tento predpoklad úplne vyvrátený.

Prepletenie implementácie s testovaním taktiež spôsobuje kolísavé testovacie úsilie v čase. Model predpokladá, že sa testovanie prebieha v pravidelných časových intervaloch v rovnakej miere. Ale pri tomto type vývoja sa čas testovania odvíja od času implementácie a úsilie od robustnosti implementovanej črty.

Ako vidno, vývoj riadený testami nespĺňa viacero predpokladov na použitie modelov rastu spoľahlivosti. Úspešne by sa dali použiť jedine v prípade zaradenia ďalšej fázy testovania po ukončení implementácie, ale to by znamenalo náklady navyše. Nehovoriac o tom, že by sa to nezlučovalo s pôvodnou ideou vývoja riadeného testami, teda vytvárať testy ešte pred implementáciou.

Záver

V tejto eseji som sa zamýšľal nad vplyvov vývoja riadeného testami na spoľahlivosť softvéru. Spôsob testovania počas tohto vývoja kladie veľký dôraz na prevenciu chýb a umožňuje priebežne sledovať stav spoľahlivosti, respektíve poruchovosti softvéru.

Ďalej som na základe predpokladov, ktoré majú modely rastu spoľahlivosti a svojich vedomostí o vývoji riadenom testami usúdil, že povaha testovania tohto druhu vývoja neumožňuje presne modelovať rast spoľahlivosti. Nemyslím si však, že je to veľký problém, pretože odhadovanie spoľahlivosti na konci vývoja považujem za rovnako

účinné ako priebežné sledovanie poruchovosti softvéru počas vývoja. Pokiaľ sa poruchovosť správne reguluje, tak pri poslednom regresnom testovaní by mala byť aktuálna poruchovosť nižšia alebo rovná cieľovej poruchovosti.

Čo sa týka zabezpečenia spoľahlivosti, požujem vývoj riadený testami za účinnejší ako vývoj, kde testovanie prebieha až po dokončení implementácie. Je to najmä kvôli vysokej prevencii chýb, ktorá v konečnom dôsledku vyžaduje nižšie náklady ako spätná oprava chýb.

Použitá literatúra

1. Jalote, P., Murphy, B., Garzia, M., Errez, B. 2004. Measuring Reliability of Software Products http://www.iiitd.edu.in/~jalote/papers/MeasuringRelofProducts.pdf
2. Michael R. Lyu: Handbook of software reliability engineering. 1. vyd. New York: Computing McGraw-Hill.1996. 819 s. ISBN 0-07-039400-8
3. Far, B.: Software Reliability Engineering for Agile Software Development. In: Canadian conference on electrical and computer engineering (CCECE '07). Vancouver: BC, 2007, s. 694-697.
4. Wood, A.: Software reliability growth models: assumptions vs. reality. In: Eighth international symposium on software reliability engineering (ISSRE '97). Albuquerque: IEEE, 2005, s. 136-143.

Annotation

Testing, path to reliability

Reliability is one of the most important parameters of the software quality. Its evaluation is often underestimated and it is done in late phases of development. It is important to consider the influence of the software development method on its reliability. I describe the main tasks of reliability management this essay. I analyze test-driven development from the point of view of the reliability and I think about the advantages over the waterfall development. I describe the way it deals with the faults. Considering the assumptions of the reliability growth models I point out the extent of fulfillment by test-driven development. I assess the need of reliability modeling in this type of development.