AKO ZLEPŠIŤ KVALITU ŠTUDENTSKÝCH PROJEKTOV POMOCOU TESTOVANIA

„Dvakrát meraj a raz strihaj.“ Ľudové príslovie

Martin Detko

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

Abstrakt. *V tejto práci sa snažím nájsť príčiny nekvalitných študentských projektov a možnosti zlepšenia ich kvality. Jedným zo spôsobov zlepšenia kvality, ktoré opisujem v tejto práci je testovanie pomocou akceptačných testov. Rozoberám možnosť zverejnenia akceptačných testov pre študentov a následné spôsoby hodnotenia. Okrem toho sa snažím rozlíšiť situáciu malých zadaní, kde je nutné iba implementovať jednoduchý algoritmus, od väčších projektov, kde je nutné robiť návrh. Ďalšou možnosťou, ako zlepšiť kvalitu je zmeniť spôsob vývoja. Konkrétne rozoberám vývoj riadený testami. V tomto prípade opisujem dôvody, prečo je dobré použiť tento druh vývoja oproti klasickým prístupom a nájsť možné problémy pri jeho použití. V závere sa snažím zhrnúť dôvody, prečo študentom poskytnúť hotové testy a kedy sa ich skôr snažiť donútiť tvoriť svoje vlastné.*

Kľúčové slová: *kvalita študentskej práce, akceptačné testovanie, vývoj riadený testami*

Úvod

Jednou z najdôležitejších častí zabezpečenia kvality softvéru je práve testovanie softvéru. Testovanie sa používa na overenie splnenia funkčných požiadaviek. Obmedzením tohto prístupu však je, že prebieha v čase testovania, z čoho vyplýva, že je príliš neskoro urobiť produkt kvalitnejším. Testy sú len tak dobré, ako testovacie prípady, ale môžu byť kontrolované, aby sa zabezpečilo, že všetky požiadavky budú testované vo všetkých

možných kombináciách vstupov a stavov systému. Avšak, nie všetky nedostatky sú zistené počas testovania. Testovanie softvéru zahŕňa činnosti overovania a validácie. Hlavným účelom overenia a schválenia činnosti, je zabezpečiť, aby softvérový dizajn, kód a dokumentácia spĺňala všetky požiadavky, ktoré boli od nej požadované [2].

Príprava študentov na prax

Vzhľadom na skutočnosť, že v praxi je testovanie tak potrebné, je dobré sa pozrieť, ako sú na túto situáciu pripravení študenti končiaci vysokú školu. Keďže počas štúdia sa na vysokých školách predmet ako testovanie softvérových projektov neučí, je potrebné, aby dobré testovanie bolo zabezpečené na ostatných predmetoch. Málokedy je to však pravda, lebo na projektoch sa učiteľ väčšinou sústredí na práve tú oblasť, ktorej sa predmet týka. Niekedy však samotnú funkčnosť programu neberú do úvahy, teda aj v prípade pádu aplikácie pri predvádzaní dokážu dať za zadanie plný počet. Z toho vyplýva, že študenti nemajú motiváciu spraviť projekt a testovanie dôkladne. V niektorých predmetoch sú však postupy iné a pre overenie správnosti študentských prác sa používajú akceptačné testy.

A prečo nevytvoriť akceptačné testy pre študentov?

Definícia podľa zdroja [2] hovorí o tom, že akceptačné testovanie potvrdzuje, že softvérový systém spĺňa pôvodne definované požiadavky. Tento test by nemal byť vykonávaný, kým softvér úspešne nedokončil testovanie systému. Akceptačné testovanie je používateľom spúšťaný test, ktorý používa black-box techniky na testovanie systému podľa jeho špecifikácie. Koncoví používatelia sú zodpovední za zabezpečenie, že všetky dôležité funkcie boli testované [2].

Táto definícia hovorí o akceptačnom testovaní v projektoch v biznise, v študentských zadaniach je to trochu inak. Podobnosť nastáva vtedy, keď učiteľa prirovnáme zákazníkovi, ktorý toto testovanie vykoná a overí splnenie špecifikácií zadania. Keďže mnohokrát sa jedná o zadania menšieho rozsahu, nerobí sa test systému. V niektorých prípadoch sú akceptačné testy poskytnuté aj študentovi, aby si spravil kontrolu programu predtým ako zadanie odovzdá.

Ako robím testy ja a kde vidím možnosti zlepšenia

Táto možnosť znižuje počet vyskytnutých chýb v projekte. Z vlastnej skúsenosti viem, že testovanie zadaní nerobím nikdy dôkladne a testy robím len na základnej úrovni, teda zisťujem, či mi program funguje pre základné vstupy. Následkom je, že môj program obsahuje veľa chýb.

Akceptačné testy poskytnuté učiteľom bývajú široké a snažia sa pokryť každú oblasť testovania. Vďaka tomuto komplexnému pokrytiu problému testami študent vie už pred odovzdaním, ako dobre je funkčný jeho program. V ideálnom prípade by to znamenalo, že aj keby boli študenti leniví vytvárať si vlastné testy, mali by v tomto prípade všetci plné počty bodov. Samozrejme, len ak by sa hodnotila iba správnosť programu. Toto však nie je až tak želaný výsledok pre učiteľov a ani pre ctižiadostivých študentov. V tomto prípade sa dá do hodnotenia zahrnúť aj iné vlastnosti programu, napríklad efektívnosť použitého algoritmu, kvalita kódu alebo kvalita dokumentácie.

Lenivý verzus poctivý študenti

Náš svet však nie je ideálny, a preto aj mnohí študenti, aj keď vedia, že ich riešenie nespĺňa akceptačné testy, zadanie napriek tomu odovzdajú. Dôvody môžu byť rôzne. Nedostatok času na dopracovanie, neschopnosť si program opraviť alebo aj obyčajná lenivosť, keď získajú pocit, že ich zadanie je dostatočne dobré na to, aby im to umožnilo absolvovať predmet. V týchto prípadoch majú učitelia na výber dve riešenia, buď zo splnenia akceptačných testov spravia nutnú podmienku pre odovzdanie zadania, a tým pádom programy, ktoré neprešli akceptačným testovaním sa budú považovať za neodovzdané.

Toto riešenie je vhodné hlavne pre lenivých študentov, ktorí majú dosť možností na to, aby ich zadanie prešlo akceptačnými testami. Na poctivých študentov však táto podmienka vytvára zbytočný tlak. Z vlastnej skúsenosti viem, že určitý tlak je potrebný pre dosiahnutie kvalitnej práce, ale ak sa človek dostane pod príliš veľký tlak a stres, prestáva efektívne pracovať. Preto je nutné zvážiť podľa obtiažnosti predmetu, aké podmienky študentom stanoviť.

Druhou možnosťou je znížiť ohodnotenie zadania podľa výsledkov akceptačných testov. V prípade poctivých alebo ctižiadostivých študentov je strata bodov dostatočná motivácia pre dopracovania zadania. Avšak lenivý študent sa môže uspokojiť aj s nesprávnym riešením.

Môj návrh riešenia

Osobne by som riešil túto situáciu tak, že by som akceptačné testy bral ako nutnú podmienku akceptovania testov, ale dopredu by som upozornil študentov na možnosť výnimiek, ak by prejavili dostatočnú snahu. Tento prístup by mohol zvýšiť aktivitu lenivých študentov a zníženie tlaku na študentov, ktorí sa snažia.

Ako to robili na univerzite v Brazílii

Doteraz som rozoberal vplyv akceptačných testov na zadania menšieho rozsahu. V prípade zadaní väčšieho rozsahu, kde je potrebné robiť analýzu a návrh, sa akceptačné testy zameriavajú nie na správnosť jednotlivých algoritmov, ale na kontrolu správania sa celého projektu. Mnohokrát si študenti v takýchto prípadoch pripravujú testy sami.

Avšak na univerzite v Brazílskom štáte Paraíba spravili experiment, v ktorom v predmete návrh softvéru namiesto zadania projektu poskytli študentom akceptačné testy. Celý projekt bol teda riadený akceptačnými testami (ATDD). Výhodou použitia týchto testov bola automatizácia testovania projektov pomocou nástroja EasyAccept.

Pred experimentom dostali študenti základné črty programu potrebné pre jeho vytvorenie spolu s opisom požiadaviek používateľa potrebných na jeho vytvorenie. Výsledky boli hodnotené subjektívnym názorom učiteľa. Po zavedení ATDD študenti vypracovávali projekt presne podľa akceptačných testov a splnenie všetkých testov znamenalo splnenie všetkých funkčných požiadaviek zákazníka. Nevýhodou tohto prístupu je skutočnosť, že učiteľ príprave zadania musí venovať o dosť viac času [3].

Podľa výsledkov [3] študenti v pôvodných zadaniach nie vždy presne vedeli určiť, čo zákazník potrebuje. V experimente vďaka testom vedeli ako sa ich program mal presne správať. Táto skutočnosť sa prejavila aj vo výsledku experimentu, kde študenti, ktorí pracovali s akceptačnými testami, mali lepšie výsledky.

Tento prístup jasne poukazuje na to, že ak by som chcel študentov naučiť a následne ohodnotiť ich schopnosť navrhovať softvér, rozhodne by som použil pre zadanie akceptačné testy. Táto skutočnosť však nepodlieha realite, kde je potrebné nielen navrhovať softvér ale aj analyzovať problémovú oblasť a špecifikáciu požiadaviek.

Ako naučiť študentov testovať vlastné projekty

Doteraz som rozoberal len prípady, keď učitelia poskytli študentom testy na otestovanie produktu. To však nepomôže študentov úplne pripraviť na prax. Je potrebné, aby si študenti vedeli sami, čo najlepšie otestovať riešenie.

Podľa mňa riešením skutočnosti, že študenti nemajú praktickú skúsenosť s vytváraním testov, by mohlo byť to, že časťou študentského projektu by bolo vytvoriť testy. Keďže väčšina študentov je lenivá alebo nemajú čas, tak študenti testy tvoria práve pred odovzdávaním projektov, ak vôbec nejaké vytvoria, ale otestovanie všetkých prípadov sa robí málokedy. Vývoj riedený testami (TDD), bol navrhnutý ako možné riešenie pre zlepšenie schopností študentov testovať softvér a realizovať výhody testovania. Pri vývoji testami si študent musí vytvoriť najprv testy a až po dokončení testov začne vytvárať implementáciu. Vďaka tomu sa predíde neskorému vytvoreniu testov a tým by sa mala zlepšiť aj kvalita softvéru.

Nič nie je dokonalé

Používanie TDD má v univerzitnom prostredí aj svoje nevýhody. Keďže sa jedná o nový, pre študentov neznámy postup vývoja, je nutné ich oboznámiť s týmto postupom. To si samozrejme vyžaduje odprednášanie danej problematiky ešte pred začatím projektu.

Ideálnym prípadom by bolo vytvorenie nového predmetu, ktorý by sa venoval testovaniu projektov a zadaní. Okrem vývoja riadeného testami by sa študenti na tomto predmete mohli oboznámiť s rôznymi druhmi testovania. Ďalej by tento predmet mal spolupracovať s ďalším predmetom, kde by si študenti mohli prakticky otestovať na konkrétnom projekte nadobudnuté vedomosti a zároveň vytvorené testy v projekte by tvorili časť ich hodnotenia v tomto predmete. V prípade, ak by nebolo možné vytvoriť nový predmet pre takéto účely, bolo by nutné začať s vysvetľovaním na predmete, kde študenti riešia návrh a implementáciu konkrétneho projektu. Keďže hodín na vysvetlenie takejto dôležitej časti nie je veľa, o to potrebnejšie je predmet svedomite naplánovať.

Keďže som študentom, tak zo skúsenosti viem, že študenti sa snažia si všetko uľahčiť, tak aby sa nestalo, že by študenti dorobili testy až nakoniec pred odovzdávaním projektu, bolo by nutné rozdeliť odovzdávanie na viac častí, kde by sa odovzdávanie testov nachádzalo ešte pred tvorením implementácie. Okrem toho, keďže testy sú nefunkcionálne časti projektu, študenti v nich nemusia vidieť hneď zmysel. Preto by učiteľov čakala ťažká úloha motivovať študentov, k svedomitej tvorbe testov a ak chceme, aby sa študenti zlepšovali, museli by im učitelia poskytovať spätnú väzbu o ich práci pri tvorbe testov.

Kedy s tým začať?

Ďalším dôležitým faktorom úspešného zvládnutia TDD je čas, kedy s ním budú študenti oboznámený. Ako študent inžinierskeho stupňa mám skúsenosti už s nejedným školským

projektom a dobre viem, ako ľahko sa aj v jednoduchších zadaniach vyskytne chyba. V mladšom veku som túto skutočnosť trochu podceňoval. Práve z tohto dôvodu predpokladám že vývoj a samotné testovanie bude mať väčší úspech u starších študentov. Článok [1] však nevylučuje ani možnosť použitia TDD v úvodných kurzoch programovania, kde je však o to dôležitejšie dobre naplánovať osnovy premetu.

Ak sa všetky doteraz spomenuté problémy podarí učiteľovi prekonať, tak v TDD vidím rozumný spôsob učenia testovania a spôsob zlepšenia kvality študentských projektov. Tento môj názor potvrdzujú aj mnohé štúdie [1], kde vývoj riadený testami pomáha študentom vytvárať komplexné projekty a tým ich pripraviť na prax.

Záver

Tak ako prístup tvorby akceptačných testov učiteľmi, tak aj učenie študentov testovať vlastné projekty, má svoje výhody. Rozdielnymi sú však ich prípady použitia. Pokiaľ sa chceme zamerať na návrh softvéru alebo na implementáciu jednoduchého algoritmu, je asi lepšie použiť akceptačné testy, ktoré vytvorí sám učiteľ. Predíde sa tým zbytočným chybám pri analýze požiadaviek alebo poskytne študentovi možnosť si rýchlym spôsobom nájsť chybu v algoritme. Nevýhodou však je tá skutočnosť, že študent si nevyskúša reálnu analýzu alebo príliš zlenivie a odnaučí sa si sám robiť testy.

Ak chceme študentov lepšie pripraviť na prax, je lepšie nechať študentov, nech si testy tvoria sami, či sa to týka akceptačných testov alebo potom počas vývoja sa riadiť TDD. Tento prístup je však pre študentov časovo náročnejší. Preto je na samotnom učiteľovi, aby zhodnotil náročnosť predmetu a rozhodol sa, či študentom poskytne testy dopredu a predíde tým zbytočným chybám alebo sa bude snažiť študenta vychovať k dobrému návyku testovania hneď od začiatku.

Použitá literatúra

1. Desai, Ch., Janzen, D., Savage K.: A Survey of Evidence for TestDriven Development in Academia. In: ACM SIGCSE Bulletin, Jún 2008, Vol. 40, No. 2
2. Lewis, W. E.: Software Testing and Continuous Quality Improvement. 2. vyd. 2004. ISBN 0-8493-2524-2.
3. Sauvé, J. P., Neto, O. L. A.: Teaching Software Development with ATDD and EasyAccept. In: ACM SIGCSE Bulletin - SIGCSE 08, Marec 2008, Vol. 40, No. 1

Annotation

How improve quality of student projects by testing

In this paper I try to find the causes of low quality of student projects and possibilities to improve their quality. One way to improve quality, described in this paper is to test projects with acceptance tests. I discuss the possibility of publication the acceptance tests for students and subsequent methods of evaluation. In addition, I try to distinguish the situation of small tasks, which need only implement a simple algorithm and larger projects where you need to make design. Another way to

improve quality is to change the way of development. Especially, I analyze the test-driven development. In this case, I describe the reasons why it is good to use this type of development compared to traditional approaches and to find potential problems with its use. In conclusion, I try to summarize if it is better to provide complete tests to students or trying to force them to make their own ones.