Semestrální práce

Metody zlepšení spolupráce vývojářů a testerů

Předmět: Zlepšování procesů budování IS

Zpracovali: Daria Draganova, Yauheniya Andreyuk

Obsah

Úvod

Problematika spolupráce vývojářů a testerů je dobře známa každému, kdo pracoval v tomto odvětví. I když jsou oba povolání velmi blízké a mají stejný cíl (dodání kvalitního softwaru), vývojáře a testery mají odlišný přístup. Vývojář se snaží vytvořit aplikaci a minimalizovat množství možných chyb, a tester se snaží všechny možné chyby najít. Tím pádem v hodně případech dochází ke konfliktu, neboť nikdo nerad uznává svoje chyby. Také často můžeme slyšet, že programátoři považují práci testerů za druhotnou a si stěžuji, že testeři se chlubí, jakmile najdou nějakou chybu a za osobní úspěch považuji, pokud programátor udělal hodně chyb.

Takže jak je patrné v komunikaci mezi vývojáře a testery je hodně nepochopení a vzájemná nepřízeň, kterou současné metodiky týmové práce se snaží eliminovat. Cílem naše semestrální práce je přiblížit problematiku zefektivněni spoluprací testerů a developerů a nalézt mezery, ovlivňující efektivitu práce. Dalším cílem je zjištění metod zlepšovaní komunikaci oddělení testerů a developerů, a vyber klíčových metod.

Proč zlepšovat komunikace

I když oba povolání jsou velmi blízké, často se stává, že programátoři vědí málo o testování a zároveň testeři vědí málo o specifických obtížnostech při programováni.

Během spolupráce testerů a vývojářů některé věci se neřeknou přímo - věci, jako například to, kde vývojáři programu mají určitou nejistotu, nebo například jakou část kódu trvalo napsat déle, než se očekávalo. Projednání podobných věci by mohlo být pro testera vodítkem, čemu by měl dat větší pozornost a jaké jsou hlavní rizika programů. Tyto body by mohli ovlivnit nebo i úplně změnit scénář testování. [1]

Jeden z nejpragmatičtějších důvodů spolupráci testerů a vývojářů je to, že tím pádem oni jsou schopní lépe zkoumat možné výsledky podmínek kódu, což umožnuje rychleji vybudovat model. Vždycky, když tester má možnost vidět jiný přístup řešení rizika, psaní kódu nebo programu, získává znalosti o dalším způsobu, jak vytvářet testovací scénář. Během spolupráci vývojáři se mohou naučit, jak efektivně testovat svůj kód a získat nový pohled na to, jak by mohl být použit jejich software. [1]

Vývojáři ale nejsou jediní, kteří se mohou naučit něco nového. Testeři zároveň mohou získat důkladnější pochopení aplikací. Vzhledem k tomu, že vývojáři sdíleli své znalosti základního kódu, tester může se dozvědět, jaké oblasti produktu mohou obsahovat slabé stránky a získat znalosti specifické pro danou aplikaci.

Příčiny problému “nepochopení”

Často se stává, že vývojáře nevidí software, jak jeho vidí tester. Oni většinou mají tendenci myslet na testy, které odrážejí typický způsob používání softwaru. Testeři naopak mají myšlení “A co by se stalo, jestli bych zkusil tohle…”, takže myslí spíše na testy, které by mohly způsobit selhání. A proto vývojáře během spolupráci s testeři jsou často překvapeni pokusy o to, aby se dostat kolem kontrolního bloku programu. Takže vývojářské perspektivy a techniky testování liší od perspektivy testerů. Testeři mají tendenci soustředit se na perspektivu uživatele, zatímco vývojáři mají tendenci soustředit se na to, co jim program vypráví o tom, co se děje v zákulisí. [2]

Taky se často stává, že vývojáře negativně reagují na snahu testera v podstatě “zničit”, to co on dlouho vyvíjel a dál tomu tolik energie a času. Jestli by vývojář a tester spolupracovali od začátku, bylo by možný lépe omezovat výskyt chyb a nebyla by potřeba něco ničit nebo předělovat.

Jak začít spolupráci?

Není úplně vhodné hned aplikovat nějakou z metod. Je možné začít několika malými pokusy experimentu s párováním. Před začátkem spoluprací je zapotřebí si vybrat vývojáře, který může poskytnout přehled na vysoké úrovni o architektuře produktu. Dále se musí vybrat vhodný projekt. Není vhodný testovat celou aplikaci v jedné relaci a navíc spolupráce vývojáře a testera funguje nejefektivněji při testování nových funkcí, a když oba účastníci pracují na projektu od začátku. Dalším krokem by mělo byt plánování - zjištění času testu, délky relace a definice zaměření testování. Také by se měli upřesnit cíle a byt nadefinované výstupy testovacích relací (při programování jediný vystup - kód, při testovaní výstupu je víc - zprávy o vadách, zkušební dokumentace, zkušební případy atd.). Před samotným začátkem práce by se melo používat prostředí, které je vhodné i pro vývojáře i pro testera aby testovali na jednom stroji. Melo by se ujistit, že oni mohou spolupracovat bez přerušení a mají možnost vzájemně mluvit. Po ukončeni práci by se měli vyhodnotit výsledky a melo by byt posouzeno jak úspěšné bylo testování, nebo popřípadě co by se dalo příště udělat jinak. [3]

Pro efektivní spolupráci jsou také podmínkou technické dovedností testera, potřebné pro testování a komunikaci s dalšími členy vývojového týmu. Jenom tester s dostatečnými technickými dovedností může sdílet a vývojářem úlohu, jako je například psaní automatizovaného testu.

Na začátku je možné zkusit spárovat testera s vývojářem na hodinu, aby provedli nějaké průzkumné testování. Před tím je taktéž možné požádat je, aby strávili pár minut chůzí a se domluvili na plánu. Je třeba požádat tým, aby stanovil měřitelný cíl, např. "Snížit počet chyb zjištěných v den vydání o 20% v příštím měsíci" nebo "v příštích dvou měsících se nesmí nalézt více než jedna chyba v den releasu".

Dalším způsobem začátku je navrhnout měsíční experiment: každý tester se spáruje s vývojářem o jeden den v týdnu a pracujte na jednom nebo obou z následujících krocích:

1.“Stories” ze zpožděné práce - "normální" práce vývojářů.
2. Psaní a provádění průzkumných testovacích “stories”, které již byly dodány a nasazeny do testovacích prostředí.

Metody zlepšení spolupráce

- **Metoda “Pair testing”**

Je to technika, ve které dvě osoby testují aplikaci ve stejném počítači, přičemž podle recenzí se vývojáři jsou schopní naučit něčemu i testery. Před začátkem testování developer a tester si dají schůzku s cílem zjistit zaměření a rozsah testů. Záměrem muže být zjištění vážné chyby, někdy zajištění, aby kritéria pro přijetí zákazníkem byla splněna, nebo najít chyby v nové funkci. Během této schůzky své cíle a testovací nápady se píšou na tabuli, a nechává se kopie hotových poznámek pro vlastní potřebu.

Vývojáři se takto naučí, jak efektivně otestovat svůj vlastní kód a získat nový pohled na to, jak by mohl být použit jejich software. Testeři získají důkladnější pochopení aplikací, které testují, a naučí se techniky ladění, aby zjistili příčiny defektů. Navíc testování v párech může porušit komunikační bariéry mezi vývojáři a testery a usnadnit budování týmu. [2]

- **Metoda “Brain Storming”**

Metoda spočívá v tom, že tým developerů a testerů si sednou spolu a začne se generování co nejvíce nápadů na určené téma. Dále se tento seznam musí být očištěn a zbývající úkoly se rozdělí mezi týmy a každý pracuje zvlášť. Většinou se takhle rysi unit-testy anebo testy jednotek kódu.

Podle recenze někteří vývojáři byli s tímto typem spolupráce mnohem pohodlnější. [2]

- **Metoda “Jeden tým”**

Tim je myšleno využití agilních metod, a to spočívá v tom, že rusí se bariera mezi tymy na rozdíl od vodopádových metod. Odstranění tohoto oddělení mezi testery a vývojáři nutí tým vyvíjet a testovat společně ve stejném sprintu a ve stejné iteraci. Tento koncept spolupráce, když neexistuji samostatné týmy pro rozvoj a QA, může způsobit zmatek, když firma začne provádět přechod od vodopádu k agilním praktikám.

Je prováděn vývoj založený na testování, a testeři tím pádem nemohou testovat každý „story“ zvlášť a testují více vrstev a pravděpodobně najdou problémy, které nemohou najít tím, že se zaměřují na jeden případ.

Tato praxe testerů a vývojářů, kteří se dají dohromady, také podporuje větší míru mentality "jednoho týmu", kterou vyžaduje úspěšná implementace Agilních metod. [3]

- **Metoda “Ping pong”**

Tato metoda znamená, že jedna osoba (tester) píše nový test a vidí, že tento test spadl na chybu. Dále tester o tom informuje vývojáře, a ten píše takovým způsobem, aby test prošel. Tester zase píše nový test a ten spádné na jinou chybu. Zase hned informuje vývojáře a proces se opakuje znovu. [3]

- **Metoda “Strong-style pairing”**

Tento styl je ve skutečnosti velmi podobný skutečné situaci navigátoru / řidiče v autě nebo na lodi, takže existuji 2 role. Role řidiče je v tomto případě nejkomfortnější, ale musí se nebát neznámým situacím a důvěřovat svému navigátoru.

Navigátor tedy má dvě hlavní úlohy:

1. **Uvést další instrukce řidiči v okamžiku, kdy je připraven jej provést**

Navigátor v podstatě spravuje seznam ToDo a velké detaily obrazu, aby se řidič mohl soustředit na kód, který píše.

2. **Mluvit na nejvyšší úrovni abstrakce, kterou může řidič pochopit**

Druhým úkolem navigátoru je mluvit na nejvyšší úrovni abstrakce, kterou řidič dokáže v současné době trávit.

Je odpovědností navigátorů komunikovat smysluplným způsobem a neměl by mluvit nad porozuměním řidičů. Je však také zodpovědností navigátorů, že stále zvyšuje úroveň komunikace a porozumění.

Silné párování funguje skvěle i pro průzkumné testování. Navigátor, který poskytuje nápady, má svobodu sledovat, co se děje bez nutnosti přemýšlet o klávesnici a řidič sleduje pokyny a píše kód. [3]

Závěr

Prvním cílem této práce bylo přiblížení čtenáři problematiky spoluprací testerů a developerů. Tento cíl jsme splnili v první kapitole, kde jsme uvedli důvody, proč by se komunikace mezi vývojáře a testery měla byt vylepšena.

Dalším cílem bylo nalezení mezer, ovlivňujících efektivitu práce. Daný cíl byl splněn v následující kapitole, která seznamuje čtenáře s původem vzniku problém spolupráce těchto dvou oddělení.

Hlavním cílem práce bylo zjištění metod zlepšení komunikaci oddělení testerů a developerů. Kostrou teto práce je kapitola “Metody zlepšení spolupráce” kde jsou popsané námi vybráni metody zlepšení komunikace vývojářů a testerů. Hlavní cíl práce jsme splnili v dané kapitole. Tím pádem všechny cíle teto práce povazujeme za splněné.

Seznam zdrojů:

[1] N. JOHNSON, Karen. *Improve Your Testing and Your Testers with Paired Testing.* [Online]. 27.04.2010. Dostupné z: http://www.informit.com/articles/article.aspx?p=1579370&seqNum=2

[2] Kohl, Jonathan. *Pair Testing: How I Brought Developers into the Test Lab*. [Online]. 2018. Dostupné z: http://www.kohl.ca/articles/pairtesting.html

[3] CRISPIN, Lisa. *Pairing With Developers: A Guide For Testers.* [Online]. Dostupné z: https://dojo.ministryoftesting.com/dojo/lessons/pairing-with-developers-a-guide-for-testers

[4] COLANTONIO, Joe. *Getting Testers and Developers To Work Together.* [Online]. 28.07.2016. Dostupné z: https://www.joecolantonio.com/2016/07/28/getting-testers-developers-work-together/

[5] PAYNE, Jeffery. *5 Ways to Pair Developers with Testers February.* [Online]. 2018. Dostupné z:

https://www.agileconnection.com/better-software-magazine-article/5-ways-pair-developers-testers