Mob Programming

Mob Programming vychází z myšlenky Pair Programmingu. Nespolupracuje tu však u jednoho počítače jen pár, ale rovnou celý tým. Dynamika celého procesu je stejná. Máme jeden počítač, jednu klávesnici, jeden ohromný monitor či plátno a jeden člověk píše to, co mu ostatní radí. Je to praktika, kterou používají hodně agilní týmy. Stejně jako Pair Programming může ze začátku působit nezvykle, ale zásadně posiluje multifunkčnost, týmovost a snižuje počet chyb nebo neporozumění. Zkuste to a uvidíte.

Review

Je jen pár praktik, které přinášejí rychle výsledek, a je navíc velmi snadné je nasadit. Review je jednou z nich. V podstatě říká, že cokoli děláte, měl by zkouknout ještě někdo jiný. A to jak kód, tak i testy, dokumentaci, design, analýzu. Nejčastěji týmy dělají code review, ale tato praktika jde opravdu použít univerzálně. Code review funguje tak, že jiný vývojář se podívá na to, co jste napsali, a připomínkuje kód. Stejně jako u Pair Programmingu, víc hlav víc ví. A dříve odhalená chyba je výrazně levnější na opravu.

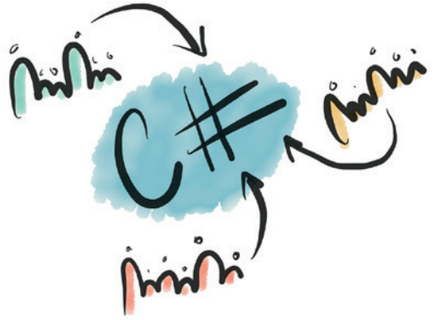
Review dělá člen týmu. Libovolný. Když je to senior či expert na danou oblast, pravděpodobně problém odhalí sám. V případě, že se naopak bude jednat o juniora, potom ten, kdo kód psal, mu změny musí vysvětlit tak, aby je pochopil a potvrdil, že je vše funkční. Tím se tím velice rychle učí a znalost o oblastech, kterou dříve měl jen jeden člen týmu, se velice rychle dostává i k ostatním členům.

Na projektech, kde je kvalita hodně důležitá až kritická, se často na každou změnu dělá ještě architecture review, kde se na všechny změny, které týmy na daném produktu udělají, ještě dívá architekt. Samozřejmě je na něm, aby si vybral klíčové změny a s minoritními drobnostmi, jež produkt nemůžou ohrozit, neztrácel čas. Kritické projekty, např. v oblastech sw pro lékaře a nemocnice, dělají ještě lead review, což je v podstatě procesní review kontrolující, že všechny změny prošly review a všechny případné review připomínky byly zapracovány, všechny předepsané testy prošly bez chyb a změna byla v této podobě akceptovaná.

Sdílený kód

Jedním z hlavních pilířů agilního programování je sdílený kód. Co to v praxi znamená? Nikdo není vlastníkem žádného kusu kódu, ať už je to specifická agenda, nebo logický celek systému. Každý člen týmu může dělat změny kdekoliv. Aby to fungovalo, je dobré tuto praktiku doplnit review a za klíčové oblasti definovat tzv. garanta, který je zodpovědný za všechny změny v dané oblasti, a všechny takové změny jdou na architecture review. Tím limitujete riziko a zároveň odstraníte slabé místo procesu.

Je to ohromná změna proti obvyklým silům, kde týmy čekají, až bude mít tento majitel sila, řekněme Karel, čas a změnu naimplementuje. Občas to funguje, občas ne. Když se firem zeptáte „A co když Karel náhle onemocní?“, obvykle s nervózním úsměvem odpovídají: „No to nesmí…“ Sdílený kód sila odstraňuje. Každý najednou může dělat změny ve mzdách. Pokud neví, pořád je tu Karel, a tak není problém se ho zeptat. A jestliže někdo z týmu něco přehlédne, Karel na to přijde během review.



Firmy se tohoto přístupu často bojí, ale praxe ukazuje, že není čeho. Nejtěžší je na tom změna myšlení lidí v týmu. Aby akceptovali, že takhle zkusí pracovat. Jakmile to zkusí, velice rychle zjistí, že černé scénáře zhroucení celého produktu nenastaly. Někteří majitelé sila sice přestanou být nepostradatelní, což může ze začátku způsobovat v jejich očích problém, ale velice rychle i oni zjistí, že jejich pozice je možná ještě lepší. Museli se sice vzdát sila, ale týmy si jich najednou více váží a zvýšil se jejich kredit. Jsou užiteční spoustě lidí. A překvapivě mají méně stresu. Nenesou tíhu celé agendy na svých zádech. Ostatní jim v případě potřeby pomůžou.

Coding Standard

Coding Standard je stejný styl pro všechny. Na začátku se to zdá být zbytečné. Proč mám pojmenovávat proměnné zrovna takhle? Proč musím používat jednu konvenci zápisu, a ne jinou? Ale po chvíli práce na větším projektu to oceníte. Kód je najednou přehledný i pro ty, kteří ho nepsali. A protože jedním ze základních pilířů agilního programování je to, že máte sdílený kód a všichni můžou dělat změny všude, je tato praktika velmi užitečná. Coding Standard nemusí mít 100+ stránek. Měl by být ale akceptovaný týmem a dodržovaný za všech situací. I když jeho nedodržení je jediným důvodem pro neakceptaci User Story v daném Sprintu.

Úspěšné týmy považují porušení Coding Standardu za důvod pro vrácení kódu z review. Z praxe můžeme říct, že to byla ze začátku hodně otravná praktika, působilo to zbytečným dojmem. Ale zpětně viděno, tým by se stal v dlouhodobém horizontu výrazně pomalejším,