VLASTNÍCTVO KÓDU A JEHO PRIMERANÉ HRANICE

Vlastníctvo kódu – dobrý sluha, zlý pán.

Daniel Petráš

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

Abstrakt. *V každom vývojárskom tíme musia platiť isté pravidlá, podľa ktorých sa tím riadi. Existuje mnoho takýchto pravidiel a je na manažmente, aby zvolil tie, ktoré prácu v tíme čo najviac zefektívnia. Takéto pravidlá nazývame organizačné vzory. Jedným z týchto vzorov je aj „vlastníctvo kódu". Zaoberám sa tým, kedy je vhodné tento vzor použiť, aké sú jeho výhody / nevýhody a riziká s ním spojené. Sumarizujem, čo všetko tento organizačný vzor zahŕňa (nie je to len o tom, kto vlastní kód). Uvádzam, prečo nie je vhodné tvrdé dodržiavanie vlastníctva kódu, a prečo je správcovstvo kódu vo väčšine prípadov vhodnejší a flexibilnejší vzor. Na záver eseje spomínam aj iné vzory a anti-vzory, ktoré s témou vlastníctva kódu súvisia.*

Kľúčové slová: *vlastníctvo kódu, vývoj v tíme, organizačné vzory*

Úvod

Väčšina vývojárov vie, čo sú návrhové vzory. Sú to vzory alebo modely, ktoré riešia istý špecifický problém na úrovni programovacieho jazyka. Každý asi pozná Model-View-Controller alebo Double-Dispatch. Viac o návrhových vzoroch sa môžete dočítať v [1]. V tejto eseji sa ale zaoberám inými vzormi, tzv. organizačnými vzormi (Organizational Patterns). Tieto vzory, na rozdiel od návrhových vzorov, riešia problém na úrovni organizácie ľudí, kolegov, alebo všeobecne tímu.

To, že v tíme použijeme niektorý zo systémov na správu verzií súborov je samozrejmé. To ale znamená, že viacero ľudí bude mať prístup k súborom a bude mať možnosť ich aj meniť. Tomu, ako rozdeliť zodpovednosti a privilégiá pre správu súborov sa venuje práve táto esej.

Zameriavam sa na vzor „vlastníctva kódu“, ktorý definuje, kto je vlastníkom súboru a čo môže so súborom vykonávať, ale aj to, aké obmedzenia platia pre tých, ktorí nie sú vlastníkmi súboru. Ďalej analyzujem, aké problémy môžu pri dodržiavaní tohto vzoru nastať, ako im predísť, alebo aké iné vzory namiesto neho použiť.

Rôzne prístupy k vlastníctvu kódu

Vlastníctvo kódu zahŕňa viac vecí, ako sa na prvý pohľad môže zdať. A dokonca môže byť chápané rôznym spôsobom. Z procesov a štandardov, ktoré tímy v organizácii dodržiavajú, môže v závislosti od veľkosti a vyspelosti organizácie *vlastníctvo kódu* pokrývať:

- Čo má kód vykonávať (požiadavky)
- Ako to má kód dosiahnuť (dizajn)
- Kto vykonáva zmeny (koordinácia)
- Kedy sa vykonávajú zmeny (koordinácia, plánovanie)

Problémy môžu nastať, keď je *vlastníctvo kódu* zredukované zo spomenutých bodov na *Ako a Kto*, alebo iba *Kto*. Bežná definícia *vlastníctva kódu* znie: „Každý kód modulu v systéme je vo vlastníctve jedného z vývojárov. S výnimkou mimoriadnych okolností je dané, že kód môže meniť iba jeho vlastník.“ Tento výrok jasne definuje *Kto*, ale ostatné dôsledky vlastníctva (hlavne *Ako* a *Kedy*) sú bez ďalšieho kontextu menej zrejmé.

Vlastníctvo kódu v tejto forme (jeden vlastník) je problematické pri malých projektoch, ktoré sú pod tlakom časového plánu. V tomto prípade absencia člena tímu, aj keď iba na relatívne krátku dobu, môže spomaliť vývoj produktu. Myslím si, že v takýchto situáciách sa oplatí mať „buddy-ho“ (ako je to v párovom programovaní), ktorý pozná kód a môže mu byť zverená zodpovednosť meniť kód a informovať o tom jeho vlastníka. Takýmto preformulovaním role vlastníka dostaneme niečo, čo sa volá *správcovstvo kódu*.

Na druhej strane, ako sa môžeme dočítať v [3], projekty ktoré neaplikujú žiaden vzor vlastníctva kódu, čelia hneď niekoľkým závažným problémom. Ako najdôležitejšie spomeniem:

1. Dlhý cyklus opravovania chyb, kde sa opakovane objavuje rovnaká chyba
2. Sústavné nestíhanie časového plánu bez zjavnej príčiny
3. Nedostatočná alebo chýbajúca dokumentácia

V spomínanom článku ([3]) autor analyzuje rôzne spôsoby, ako pristupovať k vlastníctvu kódu. Aj keď opisuje až štyri prístupy (produktový špecialista, vlastníctvo subsystému, vedúci architekt, kolektívne vlastníctvo), pri prvých troch sa stále môže objaviť problém s absenciou vlastníka kódu. Pri štvrtom prístupe je zase potrebné aplikovať nasledujúce predpoklady, ktoré nemusia vyhovovať každému tímu:

- Kontinuálna integrácia
- 100 percentné testovanie pomocou unit-testov
- Záväzný štýl písania kódu

Správcovstvo kódu

Správcovstvo kódu je alternatíva ku *vlastníctvu kódu*, zdôrazňujúc to, že kód je majetkom tímu a nie jednotlivca. Člen tímu má pridelené správcovstvo nad časťou kódu. Správca má primárnu zodpovednosť nad starostlivosťou o kód, s pomocou vstupov a rád od ostatných členov tímu. Je to správca, kto za bežných okolností vykonáva všetky zmeny v kóde, aj keď dôveryhodní členovia tímu môžu tiež vykonať zmenu v kóde, za ktorej preverenie je potom zodpovedný správca.

Tento prístup rieši problém s nedostupnosťou vlastníka kódu a zároveň nevyžaduje aplikáciu žiadnych ďalších prístupov, ktoré priamo nesúvisia zo správcovstvom kódu. Tímy ktoré príjmu za svoje ideu, že kód je vlastníctvom tímu, majú medzi sebou menej konfliktov spôsobených egom ich členov, než tímy ktoré prevezmú myšlienku vlastníctva kódu jednou osobou.

Na čo si treba dať pozor

Existuje niekoľko rizikových faktorov, ktoré sa môžu pri *vlastníctve kódu* v tíme objaviť. V eseji opisujem dve takéto riziká. Prvé sa týka nesprávneho vykonávania role vlastníka kódu. Druhé opisuje typ (personalitu) človeka, ktorý môže mať negatívny vplyv na vývoj softvéru, ak sa dostane do role vlastníka kódu.

„Razítkovacia rola“

Razítkovacia rola (rubber stamp role) je stav, kde sa vlastník kódu stáva len akousi figúrkou vo vývoji softvéru v tíme. Dochádza k tomu vtedy, keď úlohou vlastníka nie je vykonávať zmeny v kóde, ale nechá zmeny vykonávať ostatných vývojárov. Jeho úlohou je potom iba odobrenie / zamietnutie zmien v kóde.

Aj keď tento postup vo svojej podstate nie je zlý, mám pocit, že sa pri ňom dá ľahko skĺznuť do roviny, kde vlastník stráca prehľad o vlastnom kóde a odobrovanie zmien sa stáva iba administratívnou rutinou. Ďalší z dôvodov, prečo môže dochádzať k takejto degradácii role vlastníka, je snaha tímu o udržanie fikcie prehliadky kódu, aj keď sa v tíme nenachádza z daných odvetví dostatok expertov.

Prima Donna

Na začiatok si povedzme, čo je všeobecná definícia slovného spojenia *prima donna*:

1. Hlavná speváčka v opere, alebo opernej spoločnosti
2. Veľmi temperamentný človek so zvýšenou mienkou o sebe, alebo svojej dôležitosti

Je celkom zrejmé, že sa budeme zaoberať práve druhou definíciou v poradí. Jedná sa o niekoho, kto si myslí, že je nenahraditeľný, a ako dôsledok očakáva osobitné zaobchádzanie. Nie je to tímový hráč a myslí si, že preňho pravidlá neplatia. Podrobnejšiu definíciu si môžete prečítať v [2].

Čo sa však stane ak *prima donna* dostane niečo do vlastníctva? Nikto druhý nemá povolené dotknúť sa kódu (prípadne sa naň ani pozrieť) a vlastník zdržiava vývoj aj celé

týždne, pokým ostatní naňho čakajú, kým urobí požadované zmeny. *Prima donna* miluje *vlastníctvo kódu*. Dáva mu moc, vyjednávaciu pozíciu, zámienku na záchvaty hnevu a pocit nadradenosti nad tými, ktorí ho musia žiadať o povolenie na zmenu kódu. Milujú ukazovanie prstami a obviňovanie ostatných. Naopak, ľudia ktorí dobre zapadnú do tímu sa väčšinou nestarajú o *vlastníctvo kódu* (aj keď je do istej miery potrebné, ako píšem vyššie). Berú za svoje, že ak uspejú, alebo zlyhajú ako tím, tak nie je podstatné kto je zodpovedný za ktorý súbor.

Záver

Vlastníctvo kódu patrí do skupiny organizačných vzorov. Zahŕňa okrem zjavnej otázky *Kto*, aj otázky *Čo*, *Ako* a *Kedy*. Zanedbanie týchto ostatných aspektov, alebo nedostupnosť vlastníka (na istý čas), spôsobuje problémy s organizáciou v tíme a dodržaním časového plánu projektu. Ak sa však nepoužije žiaden vzor *vlastníctva kódu*, môžu sa objaviť problémy ako opakujúce sa a dlho trvajúce odstraňovanie tých istých chýb, chýbajúca dokumentácia, alebo nesplnenie časového plánu bez zjavnej príčiny.

Navrhol som riešenie využitím *správcovstva kódu*, ako kompromis medzi *vlastníctvom kódu* a nepoužitím žiadneho organizačného vzoru. Hlavný rozdiel je v tom, že vlastníkom kódu je v tomto prípade tím a nie jednotlivec. Správca má len primárnu zodpovednosť nad starostlivosťou o kód. Pri tomto prístupe sa zároveň medzi vývojármi v tíme vyskytuje menej konfliktov.

Existujú riziká ako *razítkovacia rola* a *prima donna,* na ktoré si treba dať pri používaní vzorov vlastníctva kódu pozor. *Razítkovacia rola* popisuje nesprávne použitie vzoru, kde sa z vlastníka / správcu stáva iba administratívna figúrka. *Prima donna* popisuje osobnosť človeka v priveľkým egom, pri ktorom je použitie navrhovaných vzorov problémové. Má pocit nadradenosti nad ostatnými a vytvára nezdravé prostredie na efektívnu prácu v tíme.

Použitá literatúra

1. Buschmann, F. et al, *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley and Sons, 1996, 123-168.
2. Gaskin, J. E. *High tech prima donnas*, Inter@ctive Week, 2000, 99.
3. Martin E. Nordberg III. *Managing Code Ownership*, IEEE Softw. 20, 2003, 26-33.

Annotation

Code ownership and its reasonable limits

For each development team, there must be some rules to follow. There are many such rules and the management is to choose those, which will make work in a team more efficient. Such rules are called organizational patterns. One of these patterns is the code ownership pattern. I address when it is appropriate to use this pattern and what are its advantages / disadvantages and risks associated with it. I summarize what this organizational pattern includes (it's not just about who owns the

code). I Show why unyielding compliance with code ownership is not suitable and why code stewardship is in most cases more appropriate and flexible model. Finally, the essay also mentions other patterns and anti-patterns that the subject of code ownership is related to.