O ZAMLČANÝCH NEVÝHODÁCH MANAŽMENTU VERZIÍ

Na svete existuje mnoho užitočných vecí. Otázne je, kedy tieto užitočné veci predstavujú prekážku a kedy nie.

Michal Bystrický

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
dash@serverko.sk

Abstrakt. *Manažment verzií nám a v tíme vo všeobecnosti každodenne uľahčuje prácu a dodáva nám pocit istoty v podobe záloh verzií. Existujú však prípady, kedy aplikovanie mágie manažmentu verzií nie je na mieste. Táto esej sa sústredí práve na tieto situácie. Rozoberám manažment verzií z rôznych hľadísk a venujem sa extrémnym situáciám. Zamyslím sa nad tým, či je vôbec manažment verzií v danej situácií potrebný alebo vhodný.*

Kľúčové slová: *manažment verzií, manažment, verzie, nevýhody, zálohy, SVN, GIT*

Úvod

Na ilustráciu životne dôležitého manažmentu verzií uvediem nasledovné príklady. Klient si objednal a zaplatil vývoj softvéru v našej firme. Po roku vývoja však nastane v miestnosti so servermi prepätie energie a všetky disky, na ktorých prebiehal vývoj, sú zničené. Žiadna iná kópia neexistuje, pretože vývoj prebiehal priamo na serveri (napríklad prostredníctvom FTP). Rok práce je stratený a niekto to musí zaplatiť. Ak by aj existovali zálohy, ktoré prebiehajú každý deň v noci, stratíme minimálne jeden deň práce, pretože posledné zálohy budú až z predošlej noci.

Zameňme prepätie energie so zamestnaním neskúseného programátora, ktorý omylom vymaže časť softvéru, strata je asi podobná ako v predchádzajúcom príklade.

Teda manažment verzií alebo softvér, ktorý manažment verzií zabezpečuje, sa nám v princípe stará najmä o zálohy a rozdelenie softvéru na verzie [6].

Môžeme pokračovať ďalej. Ak zlyhá manažment komunikácie a dvaja programátori pracujú na rovnakej časti zdrojového kódu, môže sa stať, že jeden omylom premaže prácu druhému. To môže mať za následok stratu niekoľkých dní práce v závislosti od štruktúry projektu. V skutočnosti, v našom príklade paralelná práca ani nemusí byť možná. Ak jeden programátor potrebuje mať vo svojej konfigurácií napríklad v databáze niečo iné ako druhý programátor, môže nastať problém. Dostali sme sa k ďalšej dôležitej základnej vlastnosti softvéru manažmentu verzií, a to zlučovanie rôznych verzií zdrojových kódov [6].

Zmeny, ktoré programátori spravia v našom príklade, nie sú uložené a zdokumentované, inými slovami kód sa časom stáva neprehľadnejší. Nevieme, kto a kedy so súborom pracoval alebo ktorá chyba bola kedy opravená. Manažment verzií nenásilným spôsobom núti programátora vytvárať zmysluplné komentáre k práci, ktorú vykonal [7]. Takto môže byť programátor kontrolovaný a teda aj motivovaný k lepším výsledkom.

Pohľad na manažment verzií

Existujú dve koncepcie manažmentu verzií, a to centralizovaný a decentralizovaný. Centralizovaný manažment verzií (napríklad SVN) umožňuje používateľom stiahnuť si súbory, s ktorými chce pracovať a po skončení práce ich nahrať na server, čím sa vytvorí ďalšia verzia. V princípe ide o jednu databázu, na ktorú sa napája mnoho klientov. Týmto teda riešime problémy záloh a verzií, no ak máme už zálohy riešené inak, pridávame tým iba podporu verizií, a ani to nemusí byť úplne tak, pretože softvér pre zálohy často podporuje aj verziovanie. Tento spôsob manažmentu verzií má mnoho nevýhod ako napríklad nasledovné [2].

- Slabá kontrola nad verziami (všetko sa deje na serveri).
- Nutnosť pripojenia na internet.
- Pomalé riešenie (komunikácia so serverom).
- Nemožnosť pracovať so súbormi bez pridelenia práv.

Takmer všetky tieto zmeny súvisia s pripojením na server. To znamená, že ak robíme vývoj v kancelárií, kde máme lokálny server, tieto nevýhody nie sú relevantné. Navyše bezpečnostná politika vo fimách často nedovoľuje pristupovať programátorom k zdrojovým kódom z iného miesta, ako vo firme.

Na druhej strane výhoda centralizovaného manažmentu verzií je rýchlosť pri inicializácií projektu (kde sa sťahuje iba hlavná vetva, nie zmeny), čo nie je podstatné, pretože túto operáciu programátor vykonáva iba raz na začiatku projektu. A vždy môžeme prepojiť decentralizovaný model cez centralizovaný (GIT cez SVN) [2]. Pri prepojení decentralizovaného modelu s centralizovaným však vznikajú problémy pri synchronizácií, kde programátor musí napríklad v GIT ignorovať súbory v SVN a naopak, aby nenastala kolízia.

Decentralizovaný manažment verzií (napríklad GIT) funguje podobne ako centralizovaný, až na to, že klient sťahuje zo servera súbory a zmeny a vytvára si vlastný

centralizovaný manažment verzií u seba na disku [3]. To mu umožňuje rýchlejšie pristupovať a ukladať zmeny, pretože všetky zmeny sa dejú na pevnom disku a teda si robí celý centralizovaný model u seba na disku. A potom po skončení určitej časti práce zmeny nahrá na server. Ďalej sa v eseji budem venovať decentralizovanému modelu.

Pozrime sa na každodennú rutinu programátora alebo jeho pracovný tok v rámci manažmentu verzií.

- Programátor si stiahne aktuálnu verziu softvéru.
- Programátor pracuje.
- Po ukončení úlohy alebo opravení chyby nahrá novú verziu. K novej verzii napíše komentár.
- Po ukončení celku úlohy nahrá zmeny na server. K novej verzii opäť napíše komentár.
- Ak sa softvéru pre manažment verzií nepodarí zlúčiť zmeny, programátor manuálne prechádza tieto konfliktné súbory, až pokiaľ sa mu ich nepodarí úspešne zlúčiť.

Rutina obsahuje štyri kroky, kedy programátor pracuje s programom pre manažment verzií. Položme si otázku.

Ide o pomoc, alebo o prekážku?

Pri jednočlenných a malých tímoch ide jednoznačne o prekážku. Programátori musia vykonávať zbytočné operácie, ktoré ich zdržujú a ktoré im zamestnávajú myseľ. (Napríklad písať niekoľko komentárov na rôzne miesta, pričom by bolo jednoduchšie písať komentár iba jeden a to tam, kde si ho iný programátor môže prečítať.) Namiesto toho, aby sa sústredili na prácu, čím by sa zvýšila produktivita celého tímu. Niekto by mohol argumentovať, že tím by nemal zálohu, čo už v dnešnej dobe neplatí, pretože každý počítač sa automaticky zálohuje do klaudu.

Na druhej strane problém môže spôsobovať aj zložitosť technológie ako takej. Programátori sa musia naučiť princíp fungovania technológie, nové názvoslovie, veľké množstvo nových príkazov, používať nové programy, riešenia pri rôznych kolíziach a podobne. Už im viac nestačí sedliacky rozum, čo môže mať minimálne zo začiatku drastický vplyv na produktivitu vo vývoji projektu. V malej firme to môže byť až deštrukčné, a teda musí zamestnávať skúsených programátorov. Teda jednoznačne závisí od skúsenosti programátora, čo má za dôsledok drahší vývoj.

Ďalšie problémy môžu spôsobovať samotné možnosti programu pre manažment verzií, a to napríklad možnosť nasilu nahrať svoju verziu na server. Výsledok tohto počinu je, že sa prepíše štruktúra a vyhodia sa preč časti iných programátorov, ktorý na projekte pracujú [4]. Tiež si predstavme napríklad, že máme tím programátorov, ktorí často nahrávajú nové verzie na server. Jeden z týchto programátorov si stiahne najnovšiu verziu zo servera a má konflikt, ktorý vyrieši a nahrá iba svoje zmeny do svojej vývojovej vetvy a potom po skončení svojej práce aj na server. Výsledok je, že všetky zmeny od jeho predposlednej zmeny až po poslednú budú odmietnuté [5].

Z pohľadu agilnej metodiky vývoja, použitie softvéru pre manažment verzií tiež nemusí byť výhoda. Môže sa zdať, že softvér pre manažment verzií úplne zapadá do

inkrementálneho typu vývoja (na konci každého šprintu vzniká nová verzia), no ako som už spomenul, v dnešnej dobe každý moderný operačný systém umožňuje zálohovať súbory do klaudu (napr. Linux Ubuntu One) a vytvárať verzie po dátume (napríklad Mac OS TimeMachine). To znamená, že ak sa chceme vrátiť späť na určitú verziu, stačí použiť štandardné prvky operačného systému.

Tu narážame na otázku, kedy už ide o manažment verzií a kedy už nie? Môžeme nazvať vyššie uvedené základné prvky operačného systému ako softvér pre manažment verzií? V dokumentácií je napísané, že ide o softvér pre zálohovanie dát a nie pre manažment verzií. To znamená, že nie je považovaný za manažment verzií, aj keď má z časti tieto vlastnosti, a teda nejde o manažment verzií z uhlu pohľadu dokumentácie.

Vráťme sa teda k vyššie uvedeným problémom a zhrňme si situáciu. Manažment verzií poskytuje nasledovné pseudo-výhody.

- Zálohy a verzie. Prečo by som sa mal učiť, inštalovať a konfigurovať softvér pre manažment verzií, keď mám zálohovací systém, ktorý podporuje verzie priamo v počítači bez akejkoľvek námahy?
- Možnosť paralelnej práce. Je žiadúce, aby pracovalo súčasne niekoľko programátorov na rovnakom súbore? Nie je chyba v zlej demodularizácií? Podarí sa správne na konci zlúčiť zmeny? Nie je lepší spôsob uzamknúť súbor na súborovom systéme a zabrániť tak nesprávnemu zlučovaniu zmien?
- Prehľadnejší vývoj. Nestačí nám softvér pre manažment projektu (napríklad Redmine)?

V príklade vyššie som spomínal niektorých programátorov, ktorí sú motivovaní k zbytočným a častým nahrávaniam novej verzie na server, pretože ich nadriadení ich sledujú. V komentároch k verziám si vymýšľajú, čo všetko veľkolepé spravili s cieľom zapôsobiť na šéfa a uľahčiť si prácu, pri tom sú to len jednoduché úkony a zvyšok voľného času surfujú na sociálnych sieťach [9]. Z príkladu vidieť, že použitie softvéru pre manažment verzií má podstatný vplyv nielen na priamo zúčastnených programátorov a administrátorov, ale aj na ľudí v manažmente.

Ďalší z problémov softvéru pre manažment verzií je verziovanie databázy. V princípe ide o dátovú štruktúru v binárnom formáte, ktorá zaberá veľa miesta na disku. Funguje to tak, že program vytvorí výpis z tejto binárnej databázy (schému a dáta), ktorý už je ako holý text a môže ísť do softvéru pre manažment verzií [1]. Toto vyžaduje ďalšiu konfiguráciu u každého programátora.

Z pohľadu administrátora serverov je situácia mierne iná. Ak by išlo o väčšie množstvo serverov, pomocou manažmentu verzií sa dá ľahko rozdistribuovať konfigurácia z jedného hlavného servera na ďalšie servery. Pri malom množstve serverov je to opäť iba zbytočná práca, ktorá sa dá vykonať manuálne rýchlejšie. Na druhej strane administrátor musí pre používateľov manažment verzií nakonfigurovať, vytvoriť používateľské účty, spravovať a robiť podporu.

Zoberme do úvahy firmu, do ktorej prídu na pohovor dvaja uchádzači – programátori. Jeden z nich neovláda žiadny program, ani problematiku manažmentu verzií a ten druhý áno. Uchádzač, ktorý ovláda manažment verzií si uvedomuje dôsledky a výhody manažmentu vo všeobecnosti, pretože sa s tým už stretol – to znamená, že je

skúsenejší avšak to neznamená, že jeho výkon v programovacom jazyku je nižší ako výkon druhého uchádzača. Skúsenosti a ovládanie programu pre manažment verzií sa považujú za základnú vedomosť, a teda uchádzač, ktorý neovláda túto problematiku, nemôže požadovať vyšší plat. Podľa [8] a prieskumu z internej účtovnej dokumentácie firmy Irisoft s.r.o. rozdiel medzi platom týchto dvoch uchádzačov môže byť až 5 a viac €. Ak by firma prijala uchádzača, ktorý problematiku neovláda, vystavuje sa tým určitému riziku. Každopádne na človeko-hodinu môže byť takýto programátor lacnejší až o 5 a viac €.

Čo sa týka použitia manažmentu verzií, ide o ďalšie zaťaženie programátora bez otázneho benefitu. V prípade, že sa firma rozhodne použiť manažment verzií, programátor musí denne stráviť písaním komentárov a prerušovaním svojej práce určité percento. Bez manažmentu verzií sa programátori na začiatku dohodnú, kto vykoná ktorú časť. Ak predstavuje napríklad 30 minút denne 6,5 % platu, pri plate 10 € na hodinu pôjde o sumu až 0,65 € na hodinu. Sumárne by tak firma ušetrila absenciou manažmentu verzií 800 € + 104 € = 904 € mesačne na jedného zamestnanca.

Pre každý podnik je dôležité, aby bol na trhu úspešný, čo môže zabezpečiť vytváraním pridanej hodnoty [4]. Pridaná hodnota je základným ukazovateľom produkcie v trhovom hospodárstve, od ktorej je možné odvodiť produktivitu práce [4]. Produktivita práce (v) sa všeobecne vyjadruje nasledovnými ukazovateľmi [4].

$$v = \frac{Q}{T} \tag{1}$$

Kde v rovnici (1) je Q objem produkcie a T vynaložená práca v jednotkách času alebo v počte pracovníkov [4]. Z tohto vzorca vyplýva, že keď chceme zvýšiť produktivitu práce (za predpokladu, že objem produkcie ostane nezmenený), tak musíme znížiť vynaloženú prácu v jednotkách času alebo počte pracovníkov. Keď chceme znížiť vynaloženú prácu, musíme zvýšiť efektivitu práce, ktorú dosiahneme napríklad znížením prácnosti a teda absenciou manažmentu verzií (zjednodušením postupu, ktorý ušetrí čas). Podľa [4] pridanú hodnotu na pracovníka možno zvyšovať rastom výstupov Q (množstvo softvéru) alebo vyšším zhodnocovaním vstupov T – viazanosť pracovníkov (účinnosť alebo efektivita) a teda ich efektívnejšieho využitia.

Záver

Definoval som prípady, kedy manažment verzií nie je vhodné použiť a to napríklad pri menších tímoch. Zaoberal som sa scenármi, kedy použitie softvéru pre manažment verzií môže byť katastrofa [5]. Poukázal som na problémy, ktoré vznikajú a ktoré sú niekdy už aj vyriešené prednastavene, a teda nie je potrebný softvér pre manažment verzií. Ukázal som, že v niektorých prípadoch môže byť absencia manažmentu verzií lacnejšou a účinnejšou alternatívou.

Z globálneho hľadiska pri väčších tímoch a väčších projektoch je manažment verzií nevyhnutný minimálne pre koordináciu programátorov. No však existujú aj tu problémy, ktoré treba riešiť a pre developerov je to určitá záťaž. Manažment verzií tak predstavuje pre programátorov určité usporiadanie a riadenie ich vývoja.

Je však dôležité si uvedomiť s akým tímom, v akej firme a s akým softvérom pre podporu robíme vývoj. Softvér pre manažment projektu môže tiež sčasti slúžiť na určitú

organizáciu programátorov pri vývoji. Tak isto aj štandardné riešenie záloh poskytuje určitú časť funkcionality softvéru pre manažment verzií. Preto sa nedá jednoznačne definovať, akú má manažment verzií vo všeobecnosti pridanú hodnotu.

Použitá literatúra

1. Belokosztolszki, A.: *Source Control and Databases*. URL: http://www.simple-talk.com/sql/database-administration/source-control-and-databases/, [cit. 2012-10-13]
2. Biškup, Z.: *GIT vs SVN*. URL: http://www.codeforest.net/git-vs-svn, [cit. 2012-10-10]
3. Chacon, S.: *Pro Git*. URL: https://github.s3.amazonaws.com/media/progit.en.pdf, [cit. 2012-10-13]
4. Grznár M., Šinský P., Marsina Š.: *Firemné plánovanie*, 2011, Sprint dva, 175-195, ISBN: 978-80-89393-35-0
5. Fay R.: *Avoiding Git Disasters: A Gory Story*, URL: http://randyfay.com/content/avoiding-git-disasters-gory-story, [cit. 2012-11-21]
6. Raymond E.: *Understanding Version-Control Systems*. URL: http://www.catb.org/esr/writings/version-control/version-control.html, [cit. 2012-10-8]
7. Yeates, S.: *What is version control? Why is it important for due diligence?* URL: http://www.oss-watch.ac.uk/resources/versioncontrol.xml, [cit. 2012-10-13]
8. *Platový profil – Programátor*, JOBAGENT.SK, s.r.o., URL: http://www.naseplaty.sk/platovy-profil/110-1010/Programator.html, [cit. 2012-10-13]
9. *TUC Briefing on online social networking and Human Resources*, Trades Union Congress, 2007, URL: http://www.tuc.org.uk/extras/facinguptofacebook.pdf, [cit. 2012-11-20]

Annotation

Disadvantages of control version system

The control version system is helping with backups and managing versions. However, there are many cases where usage of the control version system is a disadvantage. This essay concentrate on these particular cases. I discuss the control version system from the different perspectives and whether it is appropriate to use.