JEDNA VERZIA JE MÁLOKEDY POSTAČUJÚCA

Verziovanie prináša mnoho výhod, avšak nie je všetko zlato, čo sa blyští.

Michal Ošvát

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
mosvat@gmail.com

Abstrakt. *Verziovať, verziovať, verziovať. Tak by mohli znieť slová jedného ruského revolucionára, prenesené zo školských lavíc na vývoj softvéru. V dnešnej dobe je pri rozsiahlejších projektoch verziovanie súčasne nevyhnutnosťou, ako aj veľmi dobrým pomocným nástrojom. Nie je tomu tak len kvôli potrebe vytvárať rôzne verzie softvéru pri vývoji, ale potreba verziovať je podmienené často aj veľkým počtom programátorov, ktorí sa na vývoji paralelne podieľajú, a to nám takýto nástroj umožňuje. V eseji sa snažím odôvodniť potrebu verziovania, ďalej sú to jeho výhody, a najviac sa zameriavam na možné problémy, ktoré sa môžu pri verziovaní vyskytnúť, a to konkrétne pri používaní Subversion. Nanešťastie ich nie je málo.*

Kľúčové slová: *verziovanie, manažment verzií, subversion*

Úvod

Svet verzií je všadeprítomný. Či už ide o verziu aktuálneho operačného systému v telefóne, alebo verziu automobilu či bezpečnostného videosystému. Veľa produktov je označených určitým číslom vyjadrujúcim jeho aktuálnu verziu. V eseji sa zaoberám konkrétne verziovaním softvéru. Jonah Kagan [3] vo svojej eseji tvrdí, že by bolo zaujímavé zaviesť verziovanie aj napríklad do škôl, a bolo by tak možné sledovať prácu študentov na domácich úlohách, čo by bolo nepochybne zaujímavé.

Vývoj softvéru je v dnešnom ponímaní nepochybne veľmi široký pojem, za ktorým sa skrýva skutočne veľa. Pod verziovaním budeme rozumieť možnosť vyvíjať softvér

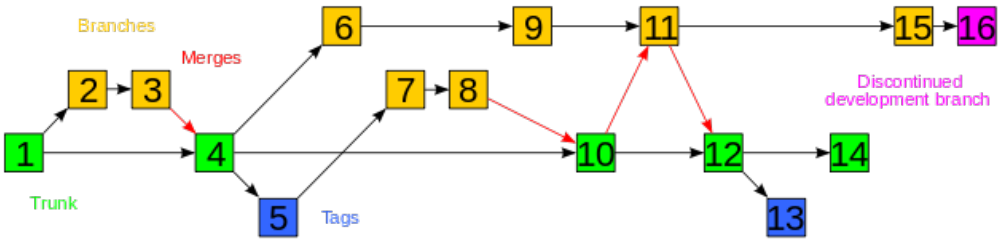
vo viacerých verziách (či už sekvenčne, alebo paralelne), pričom z toho vyplývajú nasledovné výhody:

- možnosť pre efektívnejšiu prácu viacerých ľudí nad spoločným projektom súčasne
- prehľadné sledovanie zmien
- dostupnosť a záloha všetkých pôvodných verzií
- možnosť pracovať s rôznymi verziami súčasne

Samotné verziovanie je podľa [2] súčasťou nielen podpory vývoja, ale aj podpory manažmentu. S danou myšlienkou plne súhlasím. Čo sa týka konkrétnych riešení, tak verziovanie pracuje na dvoch základných prístupoch. Buď sa používa centralizovaný prístup (napr. SVN), alebo distribuovaný prístup (napr. Git, Mercurial). Pri výbere jedného alebo druhého je potrebné zvážiť určité špecifiká, ale samotný výber vhodnejšieho prístupu na manažovanie verzií alebo prípadné porovnávanie týchto dvoch prístupov nie je predmetom tejto eseje a preto sa ním ďalej nebudem v práci zaoberať.

Subversion a jeho úskalia

Subversion (ďalej len „SVN“) je centralizovaný systém, to znamená, že obsahuje jeden centrálny repozitár, ku ktorému prostredníctvom servera pristupujú jednotliví klienti. Pre dôležitosť stručne uvediem priblíženie pojmov ako trunk (kmeň), branch (vetva) a tag (označenie, štítok). Vývoj, resp. jednotlivé verzie softvéru si môžeme pre lepšiu ilustráciu predstaviť v priestore (obr. 1) a to tak, že uzly v grafe budú predstavovať jednotlivé verzie kódu. Úplne prvotnú vetvu predstavuje trunk. Trunk je pri verziovaní vždy štartovacia línia pre projekt. Dôležité je pochopiť a uvedomiť si, že sa jedná o začiatok, od ktorého môžeme pokračovať ďalej. Pokračovanie môže predstavovať aj vytvorenie novej vetvy. Pri vytvorení vetvy dôjde k vytvoreniu kópie príslušnej verzie, ktorá ostane nezmenená, a my upravujeme ďalej už novovytvorenú vetvu. Hneď pre zjednodušenie prezradím, že tag sa od branch-u nelíši takmer v ničom. Ide len o to, že tag považujeme za určitú ucelenú verziu softvéru, ktorá môže byť odovzdaná ďalej napríklad priamo zákazníkovi. Slovom merge (spojiť) sa označuje spojenie jednej vetvy s druhou. Všetky uvedené pojmy sú ucelene znázornené na obr. 1.



Obr. 1 SVN v dvojrozmernom priestore [7].

Pomaly sa dostávam k prvým problémom, pretože jednou z nejasných a diskutabilných otázok, ktoré vznikajú pri verziovaní je:

Kedy vytvoriť novú vetvu?

Spočiatku sa nám môže zdať jednoduché a intuitívne, ale podľa [6] môže mať spôsob vytváranie vetiev pri vývoji významný dopad na celkovú kvalitu softvéru. Medzi odporúčania pri vytváraní nových vetiev patria:

- snažiť sa obmedziť vytváranie nových vetiev, pokiaľ to nie je nevyhnutné
- snažiť sa obmedziť vytváranie nových vetiev, pokiaľ bude následne viacej skupín ľudí pracovať nad príliš previazanými alebo spoločnými časťami kódu

Na vytváranie vetiev existujú konkrétne stratégie [1]:

- Branch per feature – nová vetva sa vytvorí vždy, pokiaľ ideme implementovať novú konkrétnu funkcionalitu
- Branch per iteration/sprint – nová vetva sa vytvorí vždy pre nový šprint
- Branch per Team/Subteam – projekt sa rozdelí medzi viacej tímov alebo podtímov

Treba podotknúť, že po skompletizovaní vyššie uvedených vetiev nastáva zlúčenie s pôvodnou (napr. kmeňom) alebo inou vetvou - tzv. merge. Myslím si, že uvedené odporúčania nachádzajú svoje miesto v praxi, kde sa používa verziovací systém. Pri nedodržaní určitých pravidiel sa môže celý vývoj ubrať zlým smerom, a jediným riešením bude vrátiť sa späť k poslednej funkčnej verzii a začať prácu znovu od nej, a vtedy už je celý tím v nepríjemnej situácii, keďže vynaložené úsilie (častokrát mnoho tzv. človeko-dní) na vývoj bolo zbytočné. Takýto stav je nepríjemný na jednej strane pre firmu, ktorá softvér vyvíja, pretože musí niesť následky s tým spojené, a na druhej strane taktiež pre zákazníka, keďže môže dôjsť k posúvaniu termínu dodania vyvíjanej verzie produktu.

Odovzdanie = zverejnenie kódu

Ďalším problémom pri SVN je, že nie je žiadny rozdiel medzi vytvorením a publikovaním odovzdania kódu (commit). Akonáhle niečo odovzdám, hneď je to verejne publikované pre všetkých, čo nemusí byť vždy vhodné. Pri každom odovzdávaní sa vytvorí nová revízia (revision), čo je dobré, pretože vidíme kto, kedy a akú zmenu vykonal. Ak vykonané zmeny a nie sú centrálne dostupné v repozitári, potom neboli odovzdané. Tu vstupujú do hry aj problémy so sieťovým pripojením, ktoré môžu komplikovať prácu z externej lokality. Príkladom môže byť taktiež pomalé sieťové pripojenie, ktoré môže zbytočne zaberať príliš mnoho času.

Odovzdať po každej ucelenej časti

Nevýhodu SVN vidím aj v tom, že nedokáže lokálne nahromadiť revízie, ktoré by sa potom dali naraz odovzdať. Taktiež pokiaľ programátor pracuje nad nejakým komplexnejším kódom, môže si chcieť uložiť prácu bez toho aby musel odovzdávať niečo do repozitára, a SVN nepodporuje lokálne vytváranie vetiev.

Konflikty a ich riešenie na dennom poriadku

Ďalším problémom pri verziovaní s SVN býva riešenie konfliktov. Osobne som sa stretol s riešením konfliktov hneď pri jednom z prvých kontaktov s SVN. Konflikty môžu vznikať pri odovzdávaní zmien do centrálneho repozitára, ako aj pri preberaní upravených súborov z centrálneho repozitára na lokálny disk. Rozlišujeme v podstate dva druhy konfliktov:

- *File conflict* – vzniká, ak dvaja alebo viaceri používatelia zmenia ten istý riadok alebo riadky kódu
- *Tree conflict* – vzniká práve vtedy keď sa jeden používateľ pokúša zmeniť/vymazať/presunúť súbor alebo priečinok, ktorý iný používateľ zmenil/vymazal/presunul.

Ďalej sa budem podrobnejšie zaoberať tými častejšími konfliktmi, a to sú súborové konflikty. SVN vie zistiť, že vznikol konflikt, ale vyriešiť ho automaticky nevie. Riešenie samotných konfliktov musí byť vykonávané človekom. Domnievam sa, že tento proces by sa dal len veľmi ťažko automatizovať, pretože počítač nemôže vedieť ako daný konflikt vyriešiť. Dokonca ani pri triviálnejších prípadoch riešenie konfliktu zrejmé nie je, pretože len človek vie, ktoré z ponúknutých riešení konfliktov je správne. V práci [4] autori uvádzajú, že riešenie konfliktov spolu so spájaním vetiev sú nepredvídateľné a nepraktické operácie. V práci navrhli vlastné riešenie na rozpoznávané konfliktov, avšak nepodarilo sa im vyvinúť efektívny algoritmus na automatizované riešenie konfliktov. V tejto oblasti by podľa mňa mohlo dôjsť k zlepšeniu, pretože časté rozhodovanie a riešenie konfliktov spomaľuje prácu bežného používateľa. Napriek tomu- pokiaľ je používateľ pri práci s verziovaním zručný, dobre vie na čom pracuje, tak vyriešenie konfliktov je otázkou len pár sekúnd, a práca môže pokračovať ďalej. Osobne si myslím, že problém s riešením konfliktov sa dá riešiť jednak zlepšením komunikácie v tíme, a ďalším odporúčaním je podľa mňa rozdelenie práce na projekte tak, aby konflikty nevznikali príliš často.

Zlučovanie rôznych vetiev

Ďalším problémom, na ktorý by som chcel upozorniť, je spájanie alebo zlučovanie vetiev (merge). Bryan O´Sullivan v eseji [5] uverejnenej v časopise ACM Queue (August 2009) tvrdí, že spájanie vetiev môže byť frustrujúce a nebezpečné. S týmto tvrdením taktiež súhlasím, pretože spájanie je podľa mňa kritický bod pri verziovaní. Častokrát sa pri ňom môžu znovuobjaviť už uzatvorené chyby, alebo sa vyskytnú nové. Rizikovými bodmi sú pritom hlavne:

- Vývojári, ktorí pracujú na rôznych vetvách vykonajú rôzne zmeny v tom istom súbore. Tu je obyčajne nutné vyriešenie vzniknutých konfliktov.
- Kód z jednej vetvy môže byť závislý na funkcionalite kódu z inej vetvy. Po spojení vetiev nastáva väčšinou chyba pri kompilácii.
- Veľkým problémom býva premenovanie súboru a následne jeho upravovanie v inej vetve pod starým názvom. Subversion má problémy vysporiadať sa s takýmito zmenami, preto by som neodporúčal premenovanie súborov.

Nebezpečenstvo SVN

Posledným problémom, ktorý si mnoho ľudí pracujúcich so subversion neuvedomuje, je riziko straty svojej práce ktoré čiastočne súvisí so zlučovaním vetiev. Najlepšie bude znázorniť daný problém na príklade. Na projekte pracujú dvaja programátori A a B. Subversion čísluje všetky revízie iteratívne. Predpokladajme, že aktuálna verzia je revízia č. 53. Obaja si stiahnu pomocou príkazu update aktuálnu revíziu a začnú pracovať. Po istej dobe programátor A odovzdá svoje zmeny na server, čiže vznikne revízia č.54. Neskôr chce zmeny odovzdať aj programátor B, ale Subversion mu odovzdanie nepovolí, pretože najskôr si bude musieť zlúčiť so svojim kódom revíziu č.54. Keďže programátor B má svoju prácu uloženú len lokálne, čo ak sa mu nepodarí správne zlúčenie? Jeho práca môže byť stratená alebo čiastočne poškodená. Toto riziko si mnoho ľudí neuvedomuje.

Záver

Cieľom eseje bolo priblížiť pohľad na oblasť týkajúcu sa problémov s verziovaním, konkrétne so systémom Subversion. Napriek viacerým problémom, ktoré som identifikoval, si myslím, že pri dostatočne zodpovednej a predvídavej práci môže byť SVN pre tím vhodný podporný nástroj. Pri väčšine z uvedených problémov je naznačené, ako sa im dá predchádzať. Je ale potrebné vedieť o týchto rizikách, snažiť sa im vyhnúť a správne v tíme komunikovať.

Použitá literatúra

1. Bailey, D.: When To Branch And Merge, 2010
2. Conradi, R., Wesfechtel, B.: Version models for software configuration management In: ACM Computing Surveys, Vol. 30, No. 2, (1998) 232 – 282.
3. Kagan, J.: Version Control (2012) http://www.jonahkagan.me/projects/writing/version-control.html
4. Löh, A., Swierstra, W., Leijen, D.: A principled approach to version control (2007)
5. O'Sullivan, B.: Making Sense of Revision-control System, ACM Queue (Aug., 2009)
6. Shihab, E., Bird, Ch., Zimmermann, T.: The effect of branching strategies on software quality. In Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM '12). ACM, New York, NY, USA, 301-310.
7. http://www.orsocheaffoga.altervista.org/2010/05/svn-trunk-branches-and-tags-un-minimo-di-chiarezza

Annotation

Only one version is rarely enough

Version, version, version. This is how could we modify the well-known quotation of one Russian´s revolutionary adn translate it from school environment into software development. Nowadays, a software versioning a necessity,when we consider large software projects. On the other hand it can be also very good and helpful tool. It´s not only because of a need of creating different software versions, but the need of versioning is here also because of large number of programmers, who participate parallel in software development process, and this is well supported by versioning tool. In this paper I try to substantiate the need of software versioning, discuss its advantages, and the main focus is on possible problems, which can appear, and unfortunately- there are plenty of them.