AJ KROK SPÄŤ JE NIEKEDY KROKOM K CIEĽU

Keďže zmena podmieňuje ďalšiu zmenu, verziovanie šetrí čas a peniaze.

Marek Briš

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

Abstrakt. *Vývoj softvéru je spravidla zložitý a časovo náročný proces, ktorý je v každej fáze neodmysliteľne spojený s používaním rôznych súborov či dokumentov. Nie všetky súbory sú však vo finálnej podobe, ale sú predmetom ďalšej modifikácie. Najmä práca v tíme vyžaduje možnosť modifikovania spoločných súborov, kde vykonané zmeny nepredstavujú vždy napredovanie. Často je preto potreba vrátiť sa späť, k predošlej verzii. Sledovaním vývoja softvéru z tohto pohľadu sa dnes zaoberá manažment verzií a zmien. Metódy, podľa ktorých sa dá postupovať, sú rôzne. V tejto eseji vyslovujem svoj názor na jednotlivé metódy manažmentu verzií, zdôrazňujem ich výhody a nevýhody. Zamýšľam sa nad problémami, ktoré môžu pri použití jednotlivých metód vzniknúť a aký dopad to môže mať na celkový vývoj softvéru. Som presvedčený, že používať verzie jednoducho treba, ale ktorý spôsob je ten správny?*

Kľúčové slová: *manažment verzií, systém pre správu verzií, verziovanie, prístupy k verziovaniu, version control system, vcs*

Úvod

Dnešná doba je vo veľkej miere charakterizovaná počítačmi, s ktorými je neodmysliteľne previazaný softvér. Ešte nedávno však nebol po softvéri taký dopyt ako dnes. Nároky vykladané na softvér sú omnoho väčšie ako kedysi, a preto je životný cyklus vývoja významným predpokladom úspechu finálneho softvérového produktu. Vývoj technológií a nárast zložitosti vývoja softvéru podnietil potrebu existencie podporných prostriedkov.

Základňa podporných prostriedkov je rozmanitá, kde každý z nich sa špecializuje na určitú oblasť. Jedným z pohľadov ako dozerať na vývoj softvéru je manažment verzií.

Dnes už málokto pracuje na vývoji rozsiahlejšieho softvérového produktu sám. Aj z tých, čo voľakedy začínali sami, v kuticiach svojho domu, vyrástli konkurencie schopné firmy s nemalým počtom zamestnancov. Je zrejmé, že tím zložený z odborníkov v daných oblastiach dokáže pracovať omnoho efektívnejšie a rýchlejšie ako jednotlivec. Práca v tíme si však vyžaduje zdieľanie informácií, rozdelenie úloh a aj sledovanie akým smerom a či vôbec sa vývoj softvérového produktu hýbe. Manažment verzií je v takej chvíli na mieste. Aj manažér, ktorý priamo nezasahuje do programátorských úloh, potrebuje vidieť postup práce a to prinajmenšom zmenou verzie, napr. z v1.0 na v1.1. Už len takáto skutočnosť mu dáva najavo, že veci sú v pohybe.

Ľudia, aktívne zapojení do vývoja softvéru, majú vďaka manažmentu verzií veľkú výhodu oproti iným oblastiam, ktoré majú čo to spoločné s vývojom v ktoromkoľvek zmysle slova. Veď si zoberme takého „obyčajného" maliara. Svojím spôsobom vyvíja dielo od počiatočného, bieleho plátna po krásne namaľovaný obraz. Dielo má takmer hotové, dokončuje ho a nedopatrením sa mu naň vyleje farba alebo sa inak poškodí. Tým sa jeho dlhodobá práca znehodnotí a dielo sa stane úplne nepoužiteľným. Kiežby sa mohol vrátiť v čase, pred inkriminovaný okamih, v ktorom sa jeho dielo znehodnotilo. Žiaľ, nemožné.

Verziovať??? Áno...

Cesta k finálnej podobe vyvíjaného softvéru je dlhá a kľukatá. Má rôzne zákutia, ktoré si na začiatku ani nevieme predstaviť a nad ich rozmanitosťou sa môžeme neustále čudovať. Ako sa na to pozerá manažment verzií a v čom všetkom môže byť nápomocný? Manažment verzií je postavený na podporných prostriedkoch, ktorými sú systémy pre správu verzií - VCS (angl. version control system). Je zrejmé, že správe verzií podliehajú najmä dokumenty, ktoré sú neodmysliteľnou súčasťou každého vývoja softvéru. Jedná sa o zdrojové kódy, dokumentácie, binárne súbory a iné [4]. Ako už bolo spomenuté, ďalšou výhodou je, že dokážeme usúdiť, či sa vo vývoji uberáme nejakým smerom. Či správnym alebo nie, to už je druhá vec.

Nie vždy je však nevyhnutné použiť na verziovanie systém pre správu verzií. Uvediem konkrétny príklad, ktorý sa priamo dotýkal jednej mojej známej. Ako absolventka vysokej školy sa zamestnala v jednej nemenovanej firme. Časom dostala možnosť pracovať na projekte, ktorého výstupom bola len dokumentácia. Žiaľ, vtedy nemala vedomosť o tom, že aj dokumenty je treba spravovať, zálohovať, verziovať. Len jedna, vždy aktuálna verzia dokumentu nestačila. Raz na jej počítači pracovala iná kamarátka a omylom tento dokument vymazala. Po všetkej snahe a s použitím rôznych metód sa nepodarilo vrátiť kompletný pôvodný obsah dokumentu. Trojmesačné úsilie vyšlo na zmar a nasledovala výpoveď z práce. Preto jednoznačne tvrdím, že všetko, čo sa svojím spôsobom vyvíja, treba verziovať. Hoci, nie vždy je verziovanie možné.

Podľa mňa, jedným z najdôležitejších dôvodov prečo verziovať, je možnosť vrátiť sa k niektorej z predchádzajúcich verzií. Táto možnosť v sebe zahŕňa rôzne výhody. Keďže sa jedná o vývoj softvéru, verziovaniu podlieha najmä proces implementácie. Softvér neustále rastie a stáva sa komplikovanejším a rozsiahlejším, a tímy alebo ich členovia, ktorí sa na jeho vývoji podieľajú, sú často decentralizovaní. Preto si práca v tíme vyžaduje

zdieľanie dokumentov či zdrojových súborov v rámci tímu. A tu môže dochádzať k problémom a kolíziám. Modifikovanie rovnakého súboru viacerými členmi tímu môže mať nepriaznivé dôsledky na samotnú implementáciu. Práve vtedy využijeme možnosť vrátenia sa späť, k predchádzajúcej verzii. Ďalšou výhodou je, že verziovanie nám povoľuje experimentovať. Dotknem sa opäť implementácie. Ak máme na výber viacero smerov, ktorými sa môžeme v implementácii uberať, v prípade zlyhania máme vždy možnosť opakovať voľbu a rozhodnúť sa inak. Manažér, ktorý zodpovedá za nakladanie s financiami firmy nás za to nepochváli, ale napriek tomu ho poteší fakt, že k predchádzajúcej verzii je podstatne bližšie ako na „zelenú lúku".

Čo sa týka opráv a zmien, tie musia byť starostlivo zaznamenávané a uchovávané, aby bolo možné aplikovať opravy a korekcie na mladšie či staršie časti zdrojových kódov. Dôsledok toho je, že ak vývojári používajú a hlavne veria systému na správu verzií, riskovanie spojené s robením chýb je redukované a môžu robiť radikálnejšie úpravy bez pocitu strachu [2].

Ktorou cestou sa vydať?

Ak sme sa už rozhodli pre systém pre správu verzií, máme na výber niekoľko prístupov, ktoré môžeme aplikovať. Dá sa povedať, že takmer všetky ponúkajú určitú, základnú škálu použiteľných funkcií. Taký najzákladnejší systém pre správu verzií by mal poskytovať automatickú podporu pre ukladanie, získavanie, protokolovanie a identifikáciu rôznych verzií súborov. Keď sa autor rozhodne aplikovať zmeny v systéme, okamžite sa zaznamenávajú vysvetlivky popisujúce tieto zmeny. Tieto informácie predstavujú metadáta každej aplikovanej zmeny [2]. Avšak len tieto funkcionálne danosti by už v dnešnej dobe nestačili. Preto aj systémy pre správu verzií podľahli vývoju a dnes ich už poznáme niekoľko typov. Ktorý je ten správny? Ktorý použiť v tej či onej situácií?

Centralizovaný VCS alebo „majme všetko na kope"

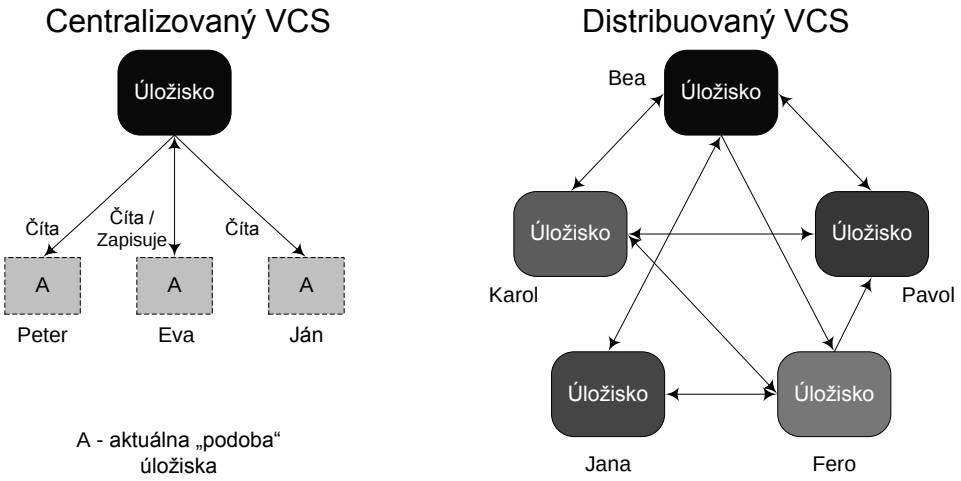
Najviac rozšírené systémy pre správu verzií používajú centralizovaný prístup, ako napríklad CVS alebo Subversion. Sú postavené na modeli klient-server. Tieto systémy sú centralizované, čiže používajú len jeden všeobecný zdrojový sklad, úložisko, ako to môžeme vidieť na Obr. 1. Vývojári pracujú výlučne nad týmto úložiskom cez tzv. *checkout* získaný zo skladu. Ide v podstate o jeho aktuálnu podobu, „snímku". Možnosť zapisovať a modifikovať úložisko je vyhradené len skupine vývojárov, tzv. *commiters*. Tímy zväčša vytvoria určité kódovacie konvencie a praktiky na kontrolu zmien nad úložiskom s cieľom chrániť kvalitu zdrojových kódov. Moderné systémy už podporujú paralelný vývoj obsahu úložiska cez vývojové vetvy (angl. branch), ktoré sa neskôr pripájajú k hlavnej vetve (angl. mainline) [2].

Áno, toto všetko je pekné, ale pozrime sa, čo všetko nám môže takýto systém priniesť. Myslím si, že výhoda ochrany úložiska a limitovanie prístupu len pre špecifickú skupinu vývojárov sa stáva skôr nevýhodou ako výhodou. Prispievatelia bez prístupu nemôžu mať prospech z centrálneho systému, a preto sa často uchyľujú k vytváraniu paralelných úložísk pre správu väčších zmien [2]. Odhliadnúc od toho, že musia byť neustále pripojení na server, je podľa mňa centralizovaný prístup úplne nevyhovujúci pri vývoji open-source projektov, pretože vtedy je právo na aktualizáciu (angl. commit) absolútne nevyhnutné.

Distribuovaný VCS alebo „každý sa hrá na vlastnom piesočku“

Alternatívou centralizovaného prístupu sa stal distribuovaný. Asi prestalo niektorých ľudí baviť, že sú odkázaní na centrálne úložisko a tým aj limitovaní. Aj Linus Torvalds, líder *linux kernel* projektu sa vyjadril, že nemá rád centralizovaný systém pre správu verzií. Distribuovaný systém je založený na modeli peer-to-peer, pretože upúšťa od požiadavky mať centrálne úložisko a tým pádom podporuje komplexnejšie interakcie. Jeho používanie umožňuje, že sa úložisko stáva opakujúce na viacerých miestach ako to môžeme vidieť na Obr.1. V neposlednom rade redukuje riskovanie niektorých katastrofických scenárov. Každý prispievateľ má vlastné úložisko a môže postupne chrániť vývoj, čo robí jeho život jednoduchším [2].

Ako už bolo spomenuté predtým, tento prístup je vhodný na open-source projekty. V prípade, že zamestnávateľ alebo vedúci vývoja vyžaduje, aby mohli zamestnanci pracovať doma, čo nie každý uvíta, je adekvátne použiť distribuovaný systém pre správu verzií.



Obr. 1. Centralizovaný vs. distribuovaný VCS.

VCS založený na príčinách

Už je nám známe, že systémy pre správu verzií poskytujú možnosť návratu k jednej z predchádzajúcich verzií v prípade poškodenia, zlyhania, napadnutia vírusom či chyby používateľa [3]. Vrátenie sa k jednej z predošlých verzií je však možné až po zásahu človeka. Ideálne by bolo, keby „niečo“ rozpoznalo zlyhanie v konkrétnej podobe a pristúpilo by k „samoobnoveniu“.

Verziovanie založené na príčinách používa vzťahy ako medzi sebou súbory súvisia. Samotný systém pre správu verzií však nedokáže sám od seba identifikovať „dobrú“ verziu súboru [3]. Kombinácia týchto dvoch pohľadov však v existujúcich systémoch často absentuje. Myslím si, že systém, ktorý svojou inteligenciou dokáže samostatne manažovať verzie v spolupráci s „ručným“ verziovaním predstavuje skutočne dobrý podporný prostriedok manažmentu verzií.

Nikdy nevyvíjaš sám

Ako už bolo spomenuté, na vývoji softvéru sa aktívne zúčastňuje tím s menším či väčším počtom členov. Vznikajúce problémy preto nie sú vôbec prekvapivé. Uvažujme nasledujúcu situáciu. V rámci jedného školského projektu som pracoval spolu s kolegom na implementácii softvérovej aplikácie. Hoci sme boli „v tíme" len dvaja, implementácia nám robila značné problémy. Keďže vtedy sme ešte nemali poznatky o existencii systémov pre správu verzií, často sa stávalo, že práca jedného bola podmienená dokončením práce druhého. Problém nastal aj pri spájaní spoločných zdrojových kódov hoci každý modifikoval inú časť. Odstupom času je táto skúsenosť úsmevná, ale aj poučná v tom, koľko trápenia nám priniesla. A to sme boli len dvaja. Čo tak tím, kde je desať, možno sto a viac vývojárov? Použitie systému, ktorý dokáže manažovať verzie, je v takom prípade nielenže potrebné, ale nevyhnutné.

Núkajú sa nám otázky či súbory zdieľať alebo nie, koľkým povoliť prístup, ako ich aktualizovať. Môžeme rozlišovať minimálne dva prístupy alebo subsystémy používané v systémoch pre správu verzií a to:

- Systém „zamykania" súborov (angl. file-locking system)
- Systém podporujúci súbežnosť prístupov (angl. concurrent system)

Pozrime sa na to, ako dokážu tieto subsystémy ovplyvniť kooperáciu medzi jednotlivými členmi tímu.

Systém „zamykania" súborov alebo „kým ťa mám"

Aj vám sa niekedy stáva, že chcete modifikovať súbor a vyskočí vám akurát systémová „hláška", že na to nemáte povolenie? Kto alebo čo to zakázalo? Toto môže postretnúť aj vývojára, ktorý má nové nápady a myšlienky, celý šťastný chce začať programovať a systém do „odfajčí". Je to spôsobené tým, že pri vývoji sa používa tzv. systém zamykania súborov. Ten dovoľuje viacerým používateľom súbory čítať, pristupovať k nim, ale len jedna osoba, šťastlivec, ho môže editovať. Podobné je to napríklad v knižnici, kde si jedna osoba knihu požičia a ostatní čakajú, kým ju vráti.

Najväčšou nevýhodou systému, ktorý povoľuje zamykanie súborov, je ich technicky vynútená tuhosť. Ak osoba, ktorá so súborom pracuje a má k nemu plný prístup je nedostupná, práca na súbore je automaticky zablokovaná, pokým nie je „zámka" prelomená. Veď stačí, keď je vývojár, ktorý daný súbor edituje, odcestovaný a pri tom nedopatrením pozabudne odomknúť súbor spolupracovníkom. Tí z toho veru šťastní nebudú. Preto si tento prístup vyžaduje veľmi dobrú koordináciu medzi členmi tímu, aby boli zosynchronizovaní a nedochádzalo k zmrazeniu súboru. Myslím si, že dokonalá synchronizácia sa však nedosahuje ľahko a skôr či neskôr k problémom či kolíziám dôjde. Podľa [1] je takýto prístup zastaralý.

Avšak na druhej strane, niekedy je primerané použiť tento systém. Kedy? No napríklad vtedy, keď sa vývojár chystá urobiť radikálne zmeny v zdrojových kódoch. V prípade, že chce niekto iný v tom istom čase používať rovnaké súbory, môže dôjsť ku kolízii. Zodpovedný vývojár s cieľom predísť katastrofickému scenáru sa zahrá na „sebca" a zamkne súbory pre ostatných.

Systém podporujúci súbežnosť prístupov

Podľa môjho názoru je v mnohých prípadoch systém „zamykania“ súborov nedostačujúci a používatelia siahnu po alternatíve. Veď čakať na to, kým sa niekomu zráči sprístupniť súbor ostatným by určite znechutilo nielen mňa. A to ešte nehovorím o tom, že po sprístupnení by ma ešte iný záujemca o ten istý súbor predbehol. Za alternatívu preto môžeme považovať systém podporujúci súbežnosť prístupov.

Ide o systém, ktorý dovoľuje viacerým vývojárom nielen pristupovať, ale i modifikovať súbor v rovnakom čase. Programovanie pod takýmto systémom sa dá považovať za zložité [1]. Keďže vývojári paralelne pracujú so súbormi dajú sa očakávať nezhody a konflikty. Najčastejšie sa stáva, že hoci vývojári súčasne pristupujú k centrálnemu úložisku, zväčša nepracujú s rovnakým súborom, ale keď už áno, tak málokedy na identickej časti. Pokiaľ ide o jednoduché zmeny, systém automaticky bez problémov zvláda zlučovanie (angl. merging) a aktualizáciu. Ak sa zmeny prekrývajú alebo sú príliš komplikované, vývojár je upozornený o konflikte a musí zmeny aplikovať manuálne. Chybám sa však nedá vyhnúť ani pri manuálnej synchronizácii [1], ba naopak. V najhoršom prípade sa dá vrátiť ku niektorej z predchádzajúcich verzií a urobiť rekonštrukciu.

Podporovanie súbežnosti prístupov v systéme pre správu verzií zvyšuje efektivitu a dokončenie práce v skoršom termíne. Dôvodom je to, že tím vývojárov, ktorý paralelne pracuje na vývoji softvéru dokáže zázraky oproti tímu, kde jeden pracuje a potenciál ostatných nie je plne využitý. Úplne najlepším riešením je podľa mňa primerané používanie oboch týchto systémov. Aplikovanie slovíčka „primerane“ v praxi si však nárokuje určitú profesionalitu vývojárov.

Záver

Manažment verzií zastáva strategickú úlohu v životnom cykle vývoja softvéru. Ako každý druh manažmentu, odráža sa od podporných prostriedkov, ktorých danosti plne využíva. Systémy pre správu verzií sa zdokonaľujú, neustále vyvíjajú a významne ovplyvňujú proces vývoja softvéru. Nakoľko nie sú totožné, zakladajú sa aj na rôznych prístupoch. Ja osobne s niektorými prístupmi súhlasím, s inými menej a zdôvodňujem prečo. V rámci tímového projektu sme sa rozhodli použiť centralizovaný systém pre správu verzií. Nakoľko sa s ním ešte stále zoznamujem, objavujem výhody a nedostatky, dá sa predpokladať, že ma priama skúsenosť ovplyvní a tým moje predchádzajúce výroky buď potvrdí alebo naopak, vyvráti.

Už aj my, „obyčajní“ študenti, pri riešení školských zadaní či projektov a ich následnej implementácii máme sklony verziovať. Učíme sa, experimentujeme a radi sa vraciame k predošlým verziám. Nepoužívame na to priamo systémy pre správu verzií avšak vývoj softvéru väčšieho rozsahu si to jednoznačne vyžaduje. Prečo? Šetrí to čas a vo firmách, ktorých predmetom je produkovať softvér, aj peniaze. Badáme, že aj krok späť je niekedy krokom k cieľu, aj návrat k predošlej verzii nás môže posunúť ďalej. Škoda len, že pod verziovanie spadajú len dokumenty či súbory. Veď si predstavme, aké by to bolo, keby sa dali verziovať „nedokumentové“ činnosti či dokonca život?

Použitá literatúra

1. Bieniusa, A., Thiemann, P., and Wehr, S. 2008. The Relation of Version Control to Concurrent Programming. In *Proceedings of the 2008 international Conference on Computer Science and Software Engineering*, December 2008, vol. 3, p. 461-464.
2. de Alwis, B. and Sillito, J. 2009. Why are software projects moving from centralized to decentralized version control systems?. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, May 2009, p. 36-39
3. Muniswamy-Reddy, K. and Holland, D. A. 2009. Causality-based versioning. In *Proceedings of the 7th Conference on File and Storage Technologies*, San Francisco, California, February 2009, p. 15-28
4. Robbes, R. 2007. Mining a Change-Based Software Repository. In *Proceedings of the Fourth international Workshop on Mining Software Repositories*, May 2007, p. 15-22

Annotation

Also step back is sometimes a good move towards goal

Software development is usually complicated and time-consuming process that is inextricably linked to each stage in use of different files or documents. Not all files are in final form, but they are a subject to further modification. Especially working in a team requires an option to modification shared files, where the changes do not always progress. Often, therefore, is needed to return back to the previous version. Versioning and change management deals with monitoring of software development from this perspective. The methods, by which it can proceed, are different. In this essay I give my opinion on various methods of version management, highlighting their advantages and disadvantages. I think about problems that may be by using different methods arise and what impact it may have on the overall software development. I believe, that we have to use versions, but which way is correct?