# DISTRIBUOVANÉ SYSTÉMY NA SPRÁVU VERZIÍ – STOJÍ TO ZA TO?

*Podporné nástroje môžu niekedy byť ako vajce a sliepka – nevieme, či sa nástroj prispôsobuje procesu, alebo proces nástroju.*

*Peter Holák*

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

***Abstrakt.*** *Jednou z najširších oblastí manažmentu konfigurácie sú systémy na správu verzií. Najmä v poslednej dobe sa táto oblasť vyvíja závratným tempom. Pomerne novým trendom sú tu systémy, ktoré namiesto centrálneho repozitára využívajú distribuovaný prístup, kde každá lokálna kópia môže slúžiť ako plnohodnotný repozitár. V tejto eseji sa pozerám na rozdiely medzi týmito druhmi systémov – ako ovplyvňujú štýl práce a pre aké typy projektov sú vhodné. Zamýšľam sa nad vplyvom výberu vhodného nástroja na samotný proces vývoja a ďalšie dopady na projekt. Vychádzam najmä z vlastných skúseností s používaním rôznych nástrojov na rôznych projektoch, no snažím sa brať si niečo aj z názorov odborníkov a ľudí, ktorí nepochybne k tejto téme majú čo povedať.*

***Kľúčové slová:*** *Subversion, git, správa verzií*

## Úvod

Každý, kto niekedy v tomto desaťročí pracoval na softvérovom projekte sa určite stretol so systémami na správu verzií (version control system – VCS). Väčšinou zrejme dostal od nadriadeného alebo iného kolegu nejaký návod, ako takýto systém v danej firme používať a príliš sa tým nezaoberal.

Takto nejak sa dajú popísať aj moje skúsenosti. V práci aj v škole som používal systémom Subversion. Ten je v súčasnosti najpoužívanejším VCS vôbec. V poslednej dobe sa však čím ďalej tým viac začali rozširovať systémy, ktoré sa odkláňajú od mnohých zavedených princípov. Ich najvýznamnejšou spoločnou vlastnosťou je distribuovaný prístup k správe všetkých do nich ukladaných dát. Táto zdanlivo technická záležitosť má široké dopady na štýl a efektívnosť komunikácie a spolupráce vývojárov projektu.

## Trocha histórie

Ako už bolo spomenuté, v súčasnosti najpoužívanejším systémom na správu verzií je Subversion. Nebolo tomu tak ale vždy – poďme sa teda pozrieť na to, čo ho dostalo do tejto pozície a ako vyzerala správa verzií pred jeho nástupom.

### Od počiatku po dnes

Za prvý skutočný systém na správu verzií systém býva považovaný Source Code Control System – SCCS. Vyvinutý v Bell Labs na počiatku sedemdesiatych rokov, umožňoval prácu len so súbormi uloženými na lokálnom úložisku. S nástupom Unixu sa práve SCCS stal dominantným systémom na správu verzií na tejto platforme, až kým jeho miesto neprevzal Revision Control System, ktorý priniesol mnohé koncepty dnes považované za bežné. Stále nepodporoval celé projekty – pracoval nad individuálnymi súbormi uloženými lokálne.

V čase, keď som sa začal viac zaujímať o IT oblasť a vývoj softvéru aj ja, som sa stretol so systémom CVS – Concurrent Versions System. Nemal som vtedy ešte veľa praktických skúseností, a tak kód veľkých projektov, ku ktorému som sa dostal, pochádzal najmä z rôzneho open source softvéru. Vtedy som zistil, že väčšina projektov okrem vydávaných stabilných a rôznych nestabilných (alfa, beta, RC) verzií umožňovala aj prístup k aktuálnemu stavu zdrojového kódu práve pomocou CVS. V tej dobe bol CVS najrozšírenejší systém na správu verzií a ja som ho používal pomocou štandardného konzolového klienta.

Aktívne pri vývoji som však začal používať tento typ nástroja až s príchodom Subversion, najmä preto že dovtedy som nič podobné nepotreboval. Subversion vznikol v roku 2000 a jeho pôvodným cieľom bolo vytvoriť systém, ktorý by bol veľmi podobný CVS, no riešil by niektoré jeho nedostatky a dopĺňal pár vlastností navyše. Tento princíp možno sledovať naprieč vývojom väčšiny podobných systémov – každý nový vznikol najmä preto, že jeho autorom nevyhovovalo nič existujúce v danej dobe a mysleli si, že by správu verzií dokázali vyriešiť lepšie.

## Nástup distribuovaných systémov

Aj keď Subversion je na stále vrchu pomyselného rebríčka systémov na správu verzií, v posledných rokoch sa čím ďalej tým viac začínajú presadzovať rôzne iné systémy, ktorých prístup k problematike správy verzií sa od dovtedy zaužívaných princípov odlišuje pomerne radikálne. Dajú sa považovať za akúsi „novú generáciu“ VCS. Hlavná vlastnosť, ktorá je spoločná pre väčšinu tejto skupiny, je distribuovaný prístup k ukladaniu všetkých dát. Namiesto existencie jedného centrálneho repozitára, s ktorým sa klienti

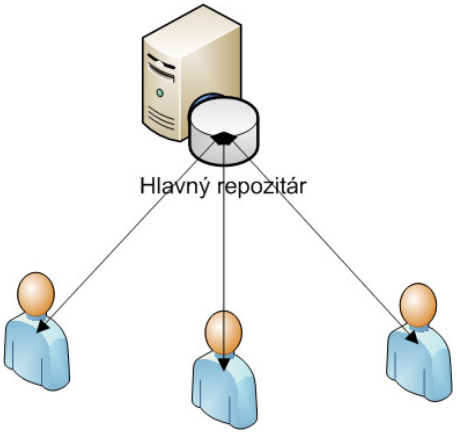
synchronizujú je každá jedna pracovná kópia u klienta sama o sebe plnohodnotným repozitárom obsahujúcim všetok obsah vrátane kompletnej histórie. Najznámejší zástupcovia sú git, Mercurial a Bazaar.

Pre človeka neznalého veci sa v súvislosti s týmto určite vyrojí množstvo otázok. Na čo je to dobré? Prečo by sme to mali používať namiesto súčasne zavedených systémov? Vieme si predstaviť význam pre určité druhy projektov, ale bude to vhodné práve pre náš projekt?

Odpovede na tieto otázky nemusia byť jednoznačné. Aj mnohí z najväčších zástancov distribuovaných systémov na správu verzií (DVCS – distributed version control system) priznávajú, že existujú prípady projektov, pre ktoré je lepšou voľbou jeden z centralizovaných systémov.

## Čo vlastne DVCS ponúkajú

Spôsob práce s centralizovaným VCS je známy asi každému, kto má nejaké skúsenosti s tvorbou softvéru – vývojári majú lokálne pracovné kópie nad ktorými vykonávajú zmeny a tieto potom posielajú do centrálneho repozitára. Zároveň si z neho sťahujú zmeny vykonané ostatnými vývojármi. Tento štýl práce je znázornený na Obr. 1. Šípky tu reprezentujú tok dát.

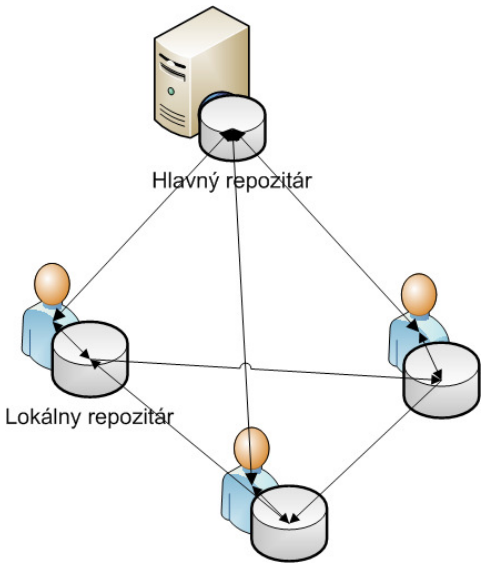



**Obr. 1.** Centralizovaný systém na správu verzií.

Naproti tomu v distribuovanom systéme nemusí existovať žiaden hlavný repozitár, aj keď v mnohých prípadoch jeden takúto úlohu plní. Takýto systém je znázornený na Obr. 2. Vývojári vykonávajú commity nad svojimi lokálnymi repozitármi, a zverejňovanie a zdieľanie kódu sa deje pomocou operácie push, ktorá posunie zmeny do iného repozitára po ich spojení s jeho aktuálnym stavom. Aj keď teda existuje možnosť mať repozitár, ktorý je považovaný za hlavný (a častokrát sa to takto používa), z hľadiska technickej stránky systému sú si všetky repozitáre rovnocenné.

Prvým a pomerne samozrejmým dôsledkom, ktorý z takéhoto prístupu vyplýva, je možnosť pracovať lokálne bez prístupu k centrálnemu repozitáru. Toto môže byť užitočné v situáciach, keď nie je dostupné pripojenie k internetu, napríklad pri cestovaní. Ja toto

nepovažujem za príliš dôležitú výhodu, no je to pravdepodobne jedna z vecí, ktoré človek ocení až v situácií, keď sa niečo také prihodí jemu. Taktiež to má za následok lepší výkon, čo sa prejaví pri častom prezeraní histórie a rozdielov. Ako ďalší dôsledok býva uvádzaná lepšia ochrana proti strate dát. Podľa mňa však pri dobre riešenom zálohovaní toto nie je problémom v žiadnom systéme.




**Obr. 2.** Príklad práce v distribuovanom VCS.

Ďalšou, a nepochybne dôležitejšou vlastnosťou je flexibilnejší spôsob práce. Pri centralizovaných VCS sa spravidla nasleduje praktika postupnej integrácie – členovia tímu dávajú svoje zmeny dokopy pomerne často pravidelnými commitmi a následne ich testujú nejakým automatizovaným nástrojom. Naproti tomu pri používaní DVCS je možné použiť mnoho rôznych štýlov práce, kde väčšinou prebieha intenzívnejšie zdieľanie kódu v rámci menších častí tímu a následne býva kód spájaný a zverejňovaný (t.j. pridaný do hlavného repozitára) vo väčších celkoch.

## Ako, kedy a prečo?

Významný princíp mnohých DVCS, predovšetkým systému git, je existencia veľkého množstva vetiev. Toto je síce možné dosiahnuť v prakticky ľubovolnom modernom VCS, no git to predpokladá už v návrhu, z čoho vyplýva robustnosť a vysoký výkon a jednoduchosť operácií týkajúcich sa tvorby a spájania vetiev.

Jeden zo štýlov práce, ktorý túto vlastnosť využíva a ktorý používa mnoho vývojárov pracujúcich s DVCS je mať jednu vetvu pre každú úlohu (branch per task). Alternatívou tohto prístupu je používanie samostatných vetiev pre všetky veľké úlohy[1]. Toto uľahčuje zoskupovanie tematicky súvisiacich zmien pre ich zverejnenie. Pre správcov hlavného repozitára to zase znamená jednoduchšiu integráciu patchov. Všetky úlohy sú vykonávané nad nejakou základnou verziou kódu a v určitom čase sú ich vetvy integrované do novej

základnej verzie. Medzitým môžu vývojári pracujúci na zložitejšej úlohe spolupracovať mimo hlavného repozitára.

Tieto výhody sa najvýraznejšie prejavia v open source softvéri, kde práva na zápis do repozitára má len úzka skupina ľudí a mnoho vývojárov nemá úroveň dôvery potrebnú na to, aby im mohla byť zverená väčšia zodpovednosť.

## Kedy je lepšie poohliadnuť sa po niečom inom

Väčšina výhod bežne uvádzaných pri distribuovaných systémoch sa naplno prejaví len pri veľkých projektoch, kde sa nedajú všetky záležitosti riešiť jednoducho osobne. Preto si napríklad neviem predstaviť, že v našom tímovom projekte by použitie DVCS prinieslo citeľnú výhodu.

Niektorí ľudia poukazujú aj na všeobecnejšie nevýhody. Ian Bicking je toho názoru, že zverejňovanie zmien skoro a často (čo centralizovaný prístup so sériovým štýlom vývoja prakticky vyžaduje) podporuje lepší spôsob práce. Iní, ako Bryan O'Sullivan však odporujú tým, že publikovanie väčších logických celkov naopak zlepšuje kvalitu komunikácie. Havoc Pennington zas poukazuje na to, že cenou za vyššiu flexibilitu DVCS je aj väčšina komplexnosť a náročnosť používania[2]. V tejto oblasti teda neexistuje univerzálna zhoda. Ja mám na základe mojich skúseností radšej situáciu, keď vidím väčšinu zmien čo najskôr, aj keď toto môže byť ovplyvnené tým, že som doteraz robil len na pomerne malých projektoch.

## Záver

Aj keď sa teda distribuované systémy na správu verzií nehodia pre každý projekt a určite centralizované systémy kompletne nikdy nenahradia, nepochybne majú čo ponúknuť a významným spôsobom ovplyvnia budúci vývoj v oblasti manažmentu konfigurácie. Nasvedčuje tomu aj fakt, že mnohé centralizované systémy začali zahŕňať niektoré vlastnosti prebraté z DVCS (aj keď samozrejme bez príliš radikálnych zmien základných princípov). Keďže väčšina DVCS vznikla len pred pár rokmi a masovo používané boli ešte kratší čas, bude zaujímavé sledovať, kam sa táto oblasť posunie počas najbližších nadchádzajúcich rokov.

## Použitá literatúra

1. Appleton, B., Berczuk, S.P., Cabrera, R., Orenstein, R.: Streamed Lines: Branching Patterns for Parallel Software Development, http://www.cmcrossroads.com/bradapp/acme/branching/branch-creation.html
2. Clatworthy, I.: Distributed Version Control Systems - Why and How, http://ianclatworthy.files.wordpress.com/2007/10/dvcs-why-and-how3.pdf
3. Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: Version Control with Subversion, http://svnbook.red-bean.com/en/1.7/svn-book.pdf

## Annotation

*Distributed Version Control Systems – Are They Worth It?*

*One of the broadest areas in software configuration management is version control systems. There has been rapid progress in this area, especially in the recent years. A relatively new trend are systems that use a distributed approach instead of a central repository. In these systems, each working copy can function as a full-fledged repository. In this essay, I look at the differences between various kinds of these systems – how they affect the work style and for what kinds of project they are suited for. I examine the effect the choice of a suitable tool has on the development process and other influences on the project. I am mostly drawing on my own experience with usage of various tools in various projects, but I am also trying to take something from the opinions of experts and people who undoubtedly have something to say about this topic.*