NAOZAJ NÁM JEDNA VERZIA NIKDY NESTAČÍ?

Občas sa aj majster tesár utne.

Tomáš Florek

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

Abstrakt. *Akýkoľvek druh softvéru, či už ide o malú voľne šíriteľnú aplikáciu, alebo rozsiahly účtovnícky informačný systém, ktorý pri svojej každodennej práci s počítačom používame, je označený číslom vyjadrujúcim jeho verziu. Za tímto často krát až smiešne zložito vyzerajúcim číslom sa však skrýva obrovské množstvo informácií, ktoré sú ale väčšine hlavne menej počítačovo gramotných ľudí neznáme. Bežný používateľ sa riadi len zaužívanou pravdou: "Čím viac tím lepšie!" Preto každú vyššiu verziu považujú za menej chybovú, jedným slovom lepšiu. Ale je to naozaj tak? Čo všetko nám toto číslo hovorí? Je vôbec nutné spravovať viac verzií? Dúfam, že sa mi aspoň trocha podarí odhaliť tajomstvá skrývajúce sa za touto problematikou.*

Kľúčové slová: *SVN, CVS, version management system*

Úvod

S pojmom verzia sa už v dnešnej dobe stretávame na každom kroku a to nie len v oblasti počítačových aplikácii. Rovnako sa stretávame s verziami mobilných telefónov, práčok, automobilov či dokonca náučnej literatúry. Väčšinou sa tieto verzie označujú kombináciou číslic, Nokia 3210, či písmen, Audi Q7, prípadne pre ľahšie odlišovanie a hlavne človeku bližšie aj slovnými pomenovaniami, Eclipse Galileo 3.5.0, a podobne. Tieto označenia nás informujú tak o technologickej vyspelosti ako aj o generácii produktu. A my automaticky predpokladáme, že každou ďalšou generáciou sa zvyšuje kvalita, spoľahlivosť a funkčnosť produktu ako aj jeho ostatné vlastnosti, ale občas je tento predpoklad ďaleko od pravdy.

Podobná situácia je aj v oblasti tvorby informačných systémov aj keď v tejto oblasti sa situácia často mierne komplikuje. Hlavne preto, že v tomto poli pôsobí veľké množstvo

vývojárov, ktorí pracujú s rôznymi štandardami a využívajú rôzne technológie a služby. Zároveň tieto nimi vyvíjané aplikácie navzájom spolupracujú, komunikujú a využívajú sa. A vzhľadom k tomu, že oblasť IT je momentálne jednou z najdynamickejších odvetví, sú všetky tieto zložky neustále zdokonaľované a menené. Avšak jednotlivé zdokonalenia prichádzajú v rozdielnych časových intervaloch a to aj v prípade systémov na sebe navzájom závislých. Aj napriek tomu, že kompatibilita takýchto symbiotických systémov je pre vývojárov jednou z priorít, sa aj majster tesár občas utne. Preto v tejto oblasti často neplatí: „Čím viac, tím lepšie." Potreba zachovania kompatibility, ale aj mnohé ďalšie dôvody si vynútili vznik systémov na monitorovanie verzií, a zmien, ktoré s nimi súvisia.

Kedy, čo a ako sa monitoruje?

Samozrejme, že s monitorovaním sa v ideálnych prípadoch začína už na začiatku projektu pri vytvorení prvého súboru, ktorý má niečo spoločné s vyvíjaným systémom. Ale rovnako je možné začať monitorovať vývoj systému v ktorejkoľvek fáze jeho životného cyklu.

Každá zmena súboru počas vývoja sa zaznamená, pričom často sú zdokumentované dôsledky a dôvody tejto zmeny. Tento proces umožňuje sledovať postup vývoja nových vlastností a funkcií ako aj identifikovať miesta vzniku chýb, čo pomáha pri ich oprave. V prípade nutnosti je možné vykonané zmeny odstrániť a vrátiť sa tak k pôvodnému stavu. Tím sa tieto systémy podobajú strojom času. V praxi som takúto udalosť pozoroval napríklad keď zapracovaná zmena, ktorá priniesla novú funkcionalitu, spôsobila príliš veľa chýb, ale v krátkej dobe bolo potrebné prezentovať projekt klientom. A vzniknuté chyby by sa nestihli opraviť.

V prípade výraznejších zmien vznikajú zálohy, ktoré sa číselne označujú a vystupujú ako verzie - tak ako ich poznáme. Ku každej verzii sa vypracúva záznam o zmenách, nazývaný tiež „changelog", tu sú zaznamenané nové funkcie, opravené chyby, ale často krát aj nové známe chyby či iné problémy napríklad ohľadom kompatibility a podobne. Preto je vhodné si tento dokument prečítať a až potom sa riadiť ľudovými múdrosťami.

Nie všetky verzie sú však zverejňované a väčšina slúži iba ako kontrolný bod v procese vývoja pre interné potreby vývojárov. Konkrétne vo firme, kde pracujem, sa takéto interné verzie ponúkajú na prezentovanie funkcií klientom a v záverečných fázach sú veľmi dôležité pri testovaní a opravovaní chýb. Ak totižto pracujete s externým tímom testerov, ako je to v našom prípade, títo dostávajú aktuálnu verziu iba pár krát týždenne, preto je informácia o tom, v ktorej verzii bola chyba objavená nesmierne dôležitá. Pomáha totižto v situáciách, keď sa nedarí zreprodukovať objavenú chybu. Toto často krát nie je spôsobené tím, že tester nesprávne opísal kroky na jej zopakovanie alebo, že by si túto chybu dokonca vymyslel. Naopak dôvodom býva zväčša to, že táto chyba bola odstránená pri prácach vykonaných od odoslania poslednej verzie do doby, keď nám bola nahlásená. Takéto chyby opravuje každý s radosťou a bleskovou rýchlosťou.

Nástroje na manažovanie verzií.

Medzi najznámejšie nástroje, ktoré vývojári používajú na činnosť spojenú s manažovaním verzií softvérových projektov, patria CVS a jeho mladší ale čoraz populárnejší brat SVN. Tieto nástroje sú zamerané hlavne na manažovanie zdrojových kódov informačných systémov a textových dokumentov. Aj preto si niektorí vývojári vytvorili podobné nástroje, ktoré boli lepšie prispôsobené ich požiadavkám hlavne na prácu zo súbormi iného charakteru ako sú obrázky, zvuky či 3D modely. Takto to bolo aj vo firme, v ktorej som začal svoju pracovnú kariéru. Nakoľko nemám s SVN a CVS žiadne osobné skúsenosti, vo svojich tvrdeniach som sa opieral o odbornú literatúru [3][4].

Spoločnou architektonickou črtou týchto systémov je prístup klient – server. Tieto môžu spolu komunikovať cez tradičnú LAN či WLAN sieť alebo prostredníctvom internetu napríklad v prípade firmy umiestnenej na niekoľkých kontinentoch či v rôznych štátoch. Tento systém teda umožňuje aj prácu z domu, keďže stačí, ak má používateľ prístup k jednému z týchto spôsobov komunikácie so serverom. Preto sú tieto systémy manažmentu verzií ideálne na prácu tak v malom ako aj vo veľkom tíme. Zároveň sú vhodné aj pre individuálne projekty, lebo si nemusíte nosiť dáta stále zo sebou.

Server slúži hlavne ako úložisko dát alebo takzvaný repozitár, kde sa ukladajú a zálohujú všetky dáta spojené s manažovanými projektmi.

Klientska strana zabezpečuje komunikáciu so serverom a vytvorenie lokálneho úložiska dát, ktoré je z väčšej časti kópiou úložiska na serveri. Toto však nemusí byť pravda pretože nie všetky súbory, ktoré u vás vznikli v súvislosti z príslušným projektom, je vhodné a potrebné skladovať v centrálnom úložisku na serveri. Všetky takéto aplikácie majú rovnakú úlohu a teda umožňujú nasledovné základné funkcie na klientskej strane aplikácie:

- Pridanie súboru na server. Umožňuje umiestniť na server prvú verziu daného súboru.
- Zmazanie súboru na serveri.
- Zapísanie aktuálnej verzie na server. Táto činnosť sa vykonáva hlavne po ukončení práce na súbore. Väčšinou sa tu uvádza aj popis vykonaných zmien. Toto sa často realizuje nahradením pôvodného súboru novým, pričom starý súbor sa zálohuje podľa nastavení systému.
- Aktualizácia lokálnych dát najnovšími, prípadne inými vyžiadanými dátami zo serveru.
- Funkcia zlúčenia verzie z lokálneho depozitára s verziou na serveri. Používa sa hlavne pri kooperatívnej editácii jedného či viacerých súborov viacerými programátormi, keď nie je možné súbor na serveri nahradiť lokálnym súborom pretože je riziko znehodnotenia kolegovej práce.
- Funkcia uzamknutia a odomknutia súboru na serveri pre ďalšiu editáciu. Využíva sa hlavne ak nechceme aby bol daný súbor upravovaný niekým iným pokiaľ na ňom pracujeme.
- Vyhľadanie rozdielnych verzií súborov na serveri a na lokálnom klientovi. Často krát sa používa na určenie toho či sme aktualizovali všetky nami modifikované súbory na strane serveru, ale rovnako aj na určenie či máme v lokálnom repozitári aktuálnu verziu produktu.

Viac privilegovaní používatelia, ktorí sú väčšinou na pozícii vedúcich tímov a projektov majú na strane servera k dispozícii aj niektoré ďalšie funkcie ako:

- Vytvorenie projektu a projektového adresára na serveri. Samozrejme, že je nutné umožniť existenciu niekoľkých projektov naraz, pretože už iba máloktorá spoločnosť vyvíja v danom čase iba jeden projekt.
- Rovnako je nutné zabezpečiť možnosť obmedzenia prístupu k projektu, pretože nie všetci pracujú na všetkých projektoch. A teda nie je potrebné a občas ani vhodné umožniť každému prístup k danému projektu.

Väčšinou sa však používajú vyššie spomenuté aplikácie CVS a SVN, keďže sú dostupné vo voľne šíriteľnej podobe a teda šetria čas, ktorý by sa spotreboval vývojom vlastnej aplikácie.

CVS vs. SVN

Ako som už vyššie spomenul SVN nazývaný tiež „Subversion" je mladším bratom CVS, avšak nie je to jediný rozdiel medzi nimi. Zatiaľ čo CVS patrí medzi predstaviteľov centralizovaného prístupu k riadeniu verzií, SVN je predstaviteľom distribuovaného prístupu. Pri tomto prístupe sa vytvára niekoľko akoby hlavných verzií, ktoré sa neskôr spájajú do finálnej verzie.

Centralizovaný prístup je určený pre užšie prepojené tímy s možnosťou osobnej komunikácie kým distribuovaný umožňuje prácu v menej zviazanom tíme, kde členovia napríklad často cestujú. Rovnako v centralizovanom systéme sú presne určené pozície v tíme a teda i prístup jednotlivých členov k súborom. Toto sa pri distribuovaných systémoch typu SVN nedá priamo obmedziť. Naopak v distribuovaných systémoch je väčšia potreba komunikovať o publikovaných zmenách zatiaľ čo v centralizovanom systéme je možné aktualizovať centrálny depozitár bez akejkoľvek komunikácie. Vetvenie verzií v centralizovanom prístupe je menej časté, málokedy sa rozvetvené verzie spájajú a ak áno je tam veľké riziko neúspechu. V distribuovaných systémoch je vetvenie a následné spájanie bežná prax. Toto vetvenie ale spôsobuje stromovú štruktúru histórie zmien čo zhoršuje prehľadnosť pri ich hľadaní. Naopak v lineárnej histórii centralizovaného prístupu sa chyby hľadajú pomerne jednoducho. Toto je len niekoľko základných rozdielov [6].

Manažment verzií v študentských projektoch.

Väčšina programátorov sa však s takýmito aplikáciami a postupmi stretáva až pri nástupe do prvého zamestnania. Je to spôsobené tím, že takýto manažment vlastnej práce hlavne na školských zadaniach považuje vzhľadom k ich veľkosti za nadbytočný a zbytočne zdĺhavý. Zároveň je to spôsobené aj systémom práce väčšiny študentov, ktorí väčšinu svojej práce, najmä pri malých typoch projektov, odkladajú na poslednú chvíľu a tak na akýkoľvek manažment neostáva čas. V tomto smere som ani ja nebol výnimkou a až niekoľko prebdených nocí ma naučilo prácu si si rozdeliť na niekoľko etáp a niekoľko deštrukčných chýb ma naučilo si jednotlivé verzie aj zálohovať. Tieto aplikácie vám pritom pomáhajú v oboch týchto smeroch. Záloha dát je automatickou súčasťou a keďže

monitorujú aj frekvenciu, množstvo a čas modifikácií nepriamo vás informujú o vašej práci na projekte.

V poslednom období sa však stáva dobrou praxou učiť študentov aj takémuto spôsobu práce či už je to v rámci predmetov ako je tímový projekt, ale aj v predmetoch kde sa tvorili aplikácie menšieho charakteru. Mnohé odborné práce pritom podporujú dôležitosť takejto snahy [1], v čom s nimi v plnej miere súhlasím. Táto snaha pomáha lepšie pripraviť absolventov na nasadenie v praxi.

Pripraviť študentov na prax však nie je jediným dôvodom nasadenia systémov pre manažovanie verzií v akademickom prostredí. Našli uplatnenie napríklad aj pri sledovaní príspevkov jednotlivých členov tímu do spoločného projektu, či sledovaní pokroku jednotlivých študentov a celého tímu [7]. Taktiež niektoré projekty poukazujú, že študenti sa často sústreďujú len na výsledok svojej práce. Pritom jednou zo snáh pedagógov je ukázať, že proces tvorby je rovnako dôležitý ako výsledok. A systémy na manažment verzií môžu pomôcť tento proces monitorovať [2]. Na druhej strane podľa záverov iných autorov ukazujú zlý vplyv využívania CVS pri vedení kurzu na pedagógove pracovné zaťaženie a efektivitu [8].

Ďalšia prípadová štúdia naopak ukazuje ako môže využívanie SVN odbremeniť pedagógov vyučujúcich podobné predmety od niektorých zdĺhavých administratívnych činností. Dobrým príkladom je najmä distribúcia materiálov ako vzorové zdrojové kódy na cvičenia a podobne. Tento systém tiež pomáhal spravovať pedagógom spoločnú internetovú stránku. Zároveň pomáhal pri interakcii so študentmi pri hodnotení ich práce, kedy boli okomentované a ohodnotené výsledky jednoducho aktualizované do centrálneho repozitára študenta. Naviac sprehľadnil a zobjektívnil hodnotenie jednotlivých členov tímu, kde bývalo bežnou praxou, že sa niektorí študenti zviezli na práci iných. Títo boli vďaka systému pomerne jednoducho identifikovaní. V neposlednom rade bola zlepšená komunikácia pedagógov medzi sebou, nakoľko mali vďaka centrálnemu repozitáru totižto prehľad aj v dokumentoch svojich kolegov. Toto všetko umožnilo pedagógom viac sa venovať pedagogickým problémom [5].

Ako vidíme tieto systémy nájdu svoje uplatnenie nielen v komerčnej sfére tímových projektov veľkých firiem ale rovnako i v akademickom prostredí. Zároveň moje skúsenosti z praxe ukazujú, že tieto systémy prinášajú veľký prínos aj v oblasti menších vlastných projektov.

Záver

V úvode som sa snažil odhaliť čitateľovi informácie skryté za magickým označením verzie, či už to boli zmeny funkcionality a vlastností, opravené chyby či známe problémy aplikácie alebo ťažkosti s kompatibilitou. Následne som ukázal, ako manažovanie verzií pomáha pri práci vývojárov, hlavne pri odstraňovaní chýb ale aj pri iných komplikáciách spojených s vykonávaním činností v tíme. Potom som sa snažil aspoň čiastočne priblížiť základnú funkčnosť systémov manažovania verzií a ich najznámejších zástupcov. V záverečnej časti som sa snažil opustiť komerčnú scénu a ponúknuť pohľad na systémy manažovania verzií v akademickom prostredí a uvietť pôsoby, ako pomáhajú vo výskumnom či pedagogickom procese. Pevne dúfam, že som zodpovedal väčšinu ak nie

všetky otázky týkajúce sa tejto témy. A ak nie, aspoň som vás motivoval prečítať si o tejto problematike viac.

Použitá literatúra

1. Beck, J.: Using the CVS version management system in a software engineering course. Journal of Computing Sciences in Colleges archive Volume 20, Issue 6 (June 2005) 57-65
2. Glassy, L.: Using version control to observe student software development proceses.(2005)
3. Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: Version Control with Subversion For Subversion 1.5, (2002),str. 1-14
4. Sigmund Support AB: Version Management with CVS (1992),str. 1-4
5. Clifton, C., Kaczmarczyk, L. C., Mrozek, M.: Subverting the Fundamentals Sequence: Using Version Control to Enhance Course Management (2007)
6. Spusta, J.: Manažment verzií alebo jedna verzia nikdy nestačí. (2009)
7. Y. Liu, E. Stroulia, K. Wong, and D. German. Using CVS historical information to understand how students develop software. In MSR 2004: International Workshop on Mining Software Repositories, 2004.
8. K. L. Reid and G. V. Wilson. Learning by doing: Introducing version control as a way to manage student assignments. In SIGCSE'05, str. 272–276. ACM Press, 2005.

Annotation

Is one version really never enough?

Every kind of software used by people in their daily life, whether it is a small personal application, or large accounting information system, is identified by its version number. This often ridiculously complicated looking numbers hides huge amount of information, however, most of which is unknown to less computer-skilled users. Regular user is restricted by the established truth: "The more, the better!" Therefore, any higher version is considered to have less errors, more features, in one word better, but is it really so? What does this number tell us? Is it necessary to manage multiple versions? I hope I will at least manage to uncover some secrets hiding behind this issue.