Verzování souborů

rešerše bakalářské práce

Osnova

- Pojmy
  - bližší probrání vybraných pojmů
- Důvody proč verzovat soubory
  - první verzovací systémy
  - vývoj systémů do dnes
- Nejběžnější verzovací systémy
  - Git
  - CVS
  - SubVersion
- Důvody tvorby nového systému
- Cíle projektu
- Zdůvodnění volby platformy
- Struktura systému
- Přínos bakalářské práce
- Závěr

Pojmy

- Branch, Fork – vývojová větev souboru, projektu, atd
- File lock – zamknutí souboru proti přepisu při checkout a odemknutí při checkin
- Checkout – vytvoření pracovní kopie souboru
- Checkin – odeslání pracovní kopie zpět do repozitáře a vytvoření revize
- Checksum, kontrolní součet – otisk dat sloužící zejména pro kontrolu neautorizovaných změn a poškození obsahu
- Mainline, Trunct – hlavní větev souboru
- Merge – sloučení více větví zpět do jedné
- Push – odeslání dat na hlavní server
- Pull – stažení aktuálních dat ze serveru
- Commit – zapsání změn do lokálního repozitáře
- Update – aktualizace pracovního adresáře z lokálního repozitáře
- Repository, Repozitář – skupina souborů a adresářů (projekt) zavedený v systému
- Revize, verze – označuje změnu v souboru nebo v jiné entitě
- Tag, label – označuje důležitou revizi ve vývoji, například stabilní verzi programu a podobně
- Workplace – pracovní adresář
- Diff – rozdíl mezi jednotlivými soubory a nebo také program, který rozdíly hledá

V následujícím textu jsou rozebrány některé pojmy, důležité pro pochopení zbytku práce.

Merge

Sloučení větví, které probíhá nejen i při současné práci více lidí na jednom souboru, je provedeno tak, že se vyhodnotí rozdíly ve slučovanách souborech a poté se vygeneruje výsledný soubor, který je sestaven ze dvou (nebo více) vstupních.

```
petr@linux-3oy1:~/nuke/temp$ diff prvni.php druhy.php
2,3d1
< $foo = "bar";
<
9c7,10
< $result = base64_encode($argv[1]);
---
> if (!empty($argv[1]))
>       $result = base64_encode($argv[1]);
> else
>       $result = "Zadal jste prázdný text"
petr@linux-3oy1:~/nuke/temp$
```

Ilustrace 1: Ukázka porovnání dvou souborů (použit program Diff)

Jako změna se bere například odebrání nebo přídání řádku textu, modifikace řádku textu, ale zpravidla je ignorována změna bílých znaků (logické odřádkování, odsazení od začátku

řádku), neboť tím se smysl textu (kódu) nemění (vyjímkou jsou ovšem programovací jazyku jako je Python, kde odsazením se uzavírají bloky kódu). Pokud je změn více u sebe, je značen jako změněný celý blok kódu (Obrázek).

Někdy nelze strojově rozdíly vyřešit (změny byly provedeny v obou souborech ve stejném bloku kódu) a je nutný zásah uživatele, aby konflikt vyřešil ručně. Tento scénář probíhá zejména při slučování dvou větví, které byly dlouho odloučené.

Branch, fork

Větvení vývoje programu může mít několik příčin. Zpravidla to bývají specifické požadavky klienta na konkrétní úpravy vyvíjeného projektu, vývoj více edic aplikace lišící se omezením funkcí a nezřítka kdy (zejména ve světě open source) i příznak odštěpení části okmunity vývojářům jako v roce 2010, kdy se od projektu OpenOffice.org oddělil fork LibreOffice.

Důvody verzování souborů

Při vývoji větších projektů (stovky a více souborů) skupinou vývojářů, ale i jednotlivcem, dochází, v případě uložení souborů projektu v klasickém adresáři na serveru, k riziku nechtěného přepisu nových souborů staršími, vzájemnému přepisování dat jednotlivými vývojáři, i k neopatrnému smazání souboru.

Hledat takovéto chyby je časově velmi náročné a hlavně značně snižuje efektivitu práce.

Řešením tohoto problému je použití některého verzovacího systému, který kromě kontroly „přepisu nových dat“ většinou zajišťuje automatické zálohování a řízení přístupu k položkám repozitáře (projektu). Veškeré operace jsou většinou logovány, tudíž zaměstnavatel může i sledovat činnost jednotlivých členů týmu.

Tyto systémy nemusí být použity pouze pro správu zdorjového kódu. Najdou použití i v širokém spektru dalších oborů, jako je správa výkresů (z praxe například propojení CADu Unigraphic a SAPu), v elektrotechnice při návrhu nových zařízení, ale i mimo technické obory, jako je školství a podobně.

Rozdělení systémů podle modelu ukládání dat

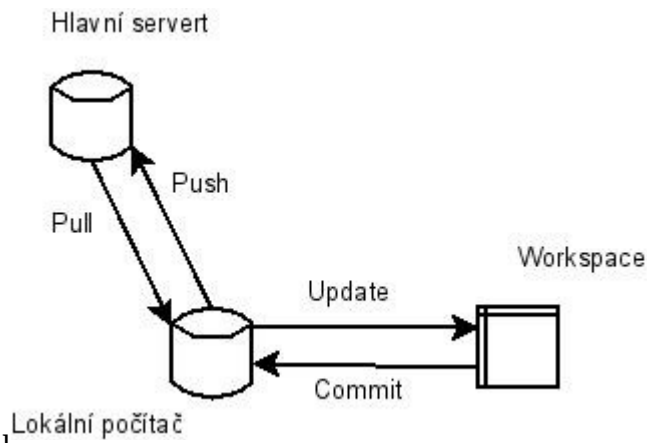
Lokální

Nejjednodušší model, kdy repozitář existje pouze na jednom stroji, kde existuje i workplace. Používá se pouze okrajově zpravidla na „one man projekty“, kdy není potřeba spolupráce s jinými vývojáři.

Model Klient-Server

V tomto případě existuje jeden centrální server, na kterém jsou všechny soubory a historie jejich změn a vývojáři mají zpravidla staženou pouze aktuální verzi souboru (projektu).

Značnou nevýhodou tohoto přístupu je, pokud se server pravidelně nezálohuje, jsou v případě havárie ztraceny všechna data.



Ilustrace 2: Model Klient-Server

Oproti tomu ale v těchto systémech můžeme soubor „zamknout“ (model Lock – Edit – Unlock) a tím zabezpečíme že v průběhu našich editací nebude moci nikdo daný soubor změnit.

Distribuované

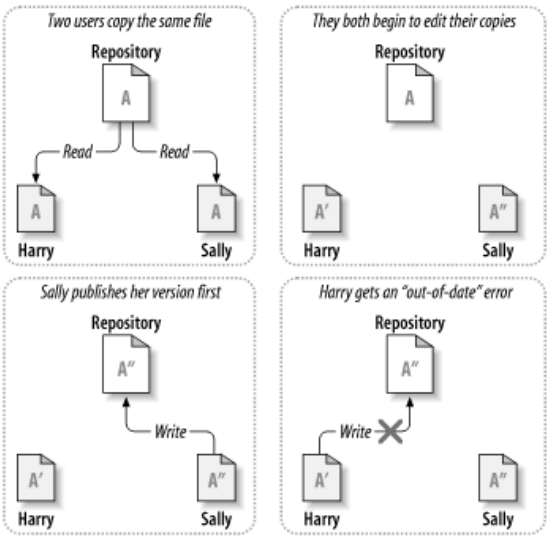
V nynější době je to nejběžnější model. V těchto systémech nemusí existovat centrální server a každý repozitář každého uživatele působí zároveň jako záloha všech dat. Z toho důvodu nelze použít model „Lock – Edit – Unlock“ (nevíme kdo dělá na kterém souboru), ale je nutné použít model, kdy je nad editovanými soubory provedeno sjednocení (merge).

Výhodou tohoto modelu je možnost práce výce vývojářů na jednom souboru zároveň, mohou pracovat nezávisle na připojení k internetu a zálohování je provedeno s každou synchronizací s uživatelem, který momentálně působí jako server.

Nevýhodou tohoto přístupu je ale posloupnost verzí, kde stejné číslo verze nemusí ještě znamenat stejný obsah souboru.

Rozdělení systému podle modelu přístupu k datům

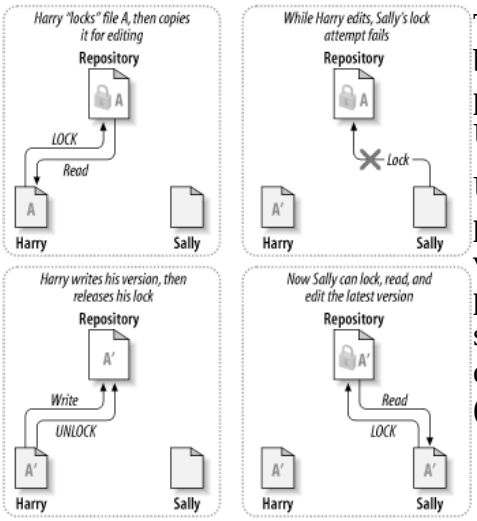
Model Zkopíruj-Modifikuj-Sluč



Nejčastěji používaný model pro verzování textově orientovaných dokumentů spočívající na filosofii slučování změn provedených ve stejnou dobu na stejném souboru.

Obrázek okazuje scénář, kdy dva vývojáři (Harry a Sally) si stáhnou jeden soubor a začnou v něm provádět změn. První z nich bezproblémů uloží změny, ale při ukládání kopie od Harryho repozitář ohlásí chybu že soubor byl od svého stažení změněn. V tomto případě je nutné sáhnout soubor se změnami od Sally a provést merge (ne vždy se tento proces obejde bez zásahu uživatele) a poté soubor konečně odeslat na server.

Model Zamkni-Uprav-Odemkni



Tento model se zpravidla používá pro správu projektů s binárními soubory, jako je projekt z CAD systému a podobně (příklad z praxe je systém SAP + CAD UniGraphic).

Uživatel, který chce soubor editovat pošle na server požadavek na zamčení souboru. V případě, že mu systém vyhoví, je mu dovoleno stáhnout požadovaný dokument pro editaci. Po uložení změn a odeslání souboru zpět na server dojde k jeho opětovnému odemčení. Po celou dobu, kdy je soubor zamčen, nesmí nidko soubor editovat (systém to nepovolí).

Z toho vyplývají rizika, se kterými je tento model spojet. Nejčastější případ chyby je pravděpodobně neodemknutí souboru po editaci a zapomenutí odeslat zeditovaný soubor zpět na server. Tento problém se dá řešit buď časovými zámky (soubor lze zamknout pouze na omezenou dobu) nebo zásahem adminstrátora, který soubor opět odemkne.

První verzovací systémy

SCCS – Source Code Controll systém

Tento systém byl vyvinut v roce 1972 v Bellových laboratořích pro korporaci IBM.

Používá velmi zvláštní architekturu pro ukládání programových jednotek (souborů), asice že veškeré verze a revize jsou uloženy v jednom souboru pomocí takzvaných delta balíčků (ukládají se pouze změny oproti předchozí verzi), které jsou řetězeny tak, jak byly postupně ukládány. Díky této architektuře není prakticky možné provést neautorizovanou změnu uloženého kódu.

Zajímavostí je, že původní soubor je reprezentován také jeho delta balíček, který je brán vzhledem k prázdnému souboru.

Po provedení každého commitu je zároveň zdokumentováno, kdo a kdy změnu provedl a zároveň je schopen provést diferenci jednotlivých revizí a zjistit ve kterých místech ke změně došlo.

Při provedení pullu je proveden proces poskládání jednotlivých delt a výsledné soubory jsou odeslány uživateli.

V současné době je tento software dostupný ve verzi 1.0, která je dostupná pro Windows a Unix like platformy. V době psaní této práce byla poslední změna stabilní verze zveřejněna na začátku roku 2007.

RCS – Revision Controll System

Jedná se dnes již značně zastaralý software, který umí pracovat s textovými dokumenty a omezeně i s binárními soubory.

Byl vyvinut vyvinut Walterem F. Tichým na Univerzitě Purdue na počátku osmdesátých let minulého století.

Je to poměrně jednoduchý systém, který dokáže pracovat pouze s jednotlivými soubory (s projekty pracovat neumí), které ukládá podobně jako SCCS pomocí delt, ale na rozdíl od něj je ukládá do samostatných souborů.

Přestože umí vytvářet a slučovat vývojové větve souboru, práce s těmito funkcionalitami je velmi obtížná, a proto se většinou pracuje pouze s mainline souboru. Tyto problémy později vyřešil systém CVS, kterým se budu zabývat později.

Ilustrace 3: Walter F. Tichy - tvůrce RCS

Poslední stabilní verze je z roku 1995 a od té doby je projekt z hlediska vývoje prakticky mrtvý. Jediná změna byla provedena na konci roku 2010, kdy byl projekt převeden na licenci GPL v3.

PVCS – Polytron Version Control system

Program byl vyvinut firmou Polytron v roce 1985. Polytron poté prošel sérií různých vlastníků a dnes je PVCS vyvíjeno firmou Serena Software Inc..

Je to komerční software, který je dnes patrně jedním z nejkomplexnějších řešení verzování projektů. Balík obsahuje sadu několika mocných nástrojů, které usnadňují práci v systému a kontrolu nad soubory a projekty.

Systém je možné ovládat jak pomocí příkazové řádky, tak i pomocí grafického rozhraní, které je dodáváno.

V momentální době je PVCS ve verzi 8.4.1, která byla vydána 29. listopadu 2010.

Nejběžnější verzovací systémy

Git

Systém začal být vyvýjen Linusem Tovaldsem v roce 2005 (projekt byl započat ve stejné době jako Mercurical) pro řízení vývoje jádra operačního systému Linux. Původně byl zamýšlen pouze jako platforma, na které se měly stavět vlastní verzovací systémy. V průběhu používání se ale Git vyvinul v plnohodnotný SMC systém, používaný s oblibou mnoha vývojáři.

Díky tomu, že se jedná o open source program, je možné ho volně využívat jak soukromě pro osobní potřebu, tak i komerčně ve velkých firmách a modifikovat ho podle svých potřeb (samozřejmě v souladu s licencí GPL).

Na serveru http://www.github.com se může kdokoliv registrovat a vytvořit se neomezený počet repozitářů.

Princip ukládání dat

Na rozdíl od většiny ostatních systémů ukládá Git vždy při každém commitu všechna data souboru znovu. To má za následek větší odolnost proti poškození repozitáře například neopatrným smazáním starých dat uživatelem nebo chybou hardwaru. Značnou nevýhodou tohoto přístupu je ovšem větší diskový prostor, který repozitář zabere. Z toho důvodu jsou staré commity komprimovány a archivovány zvlášť.

Další vlastností při ukládání dat je distribuovanost systému. To znamená, že každý vývojář má na svém stroji vlastní repozitář (kopii aktuálního repozitáře na hlavním serveru), se kterým pracuje a po ukončení práce všechny změněné soubory odešle na hlavní server. Při přenosu jsou soubory zakomprimovány a odeslány atomicky najednou, aby nedocházelo ke zbytečnému zatěžování sítě.

CVS

Původně vzniklo jako nadstavbová kolekce skriptů pro RCS a záměrem bylo zjednodušit práci s tímto systémem. V průběhu let se ale projekt vyvinul v plnohodnotný systém, který je dnes ale už relativně zastaralý a byl nahrazen níže zmíněným SubVersion.

Pracuje nad modelem Client-Server s možností takzvaného „unreserved checkout“, tedy že více vývojářů může pracovat na jedné entitě zároveň (soubor se nezamyká a vytvoří se samostatná vývojová větev pro každého vývojáře) a po odeslání na server se provede merge.

Princip ukládání dat

Metoda ukládání dat je z historických důvodů převzata z RCS a rozšířena o modulárnost datového modelu, aby u velkých projektů byla zlepšena přehlednost.

Dále systém podporuje import vývojových větví z jiných repozitářů (více týmů pracuje na stejném projektu v různých repozitářích), což je realizováno pomocí merge nad původní a importovanou branche.

SubVersion

První release byl uvolněn v roce 2000 a záměrem bylo nahradit v tu dobu nejpoužívanější, ale už značně zastaralé CVS.

Subversion používá architekturu Klient-Server a model Zkopíruj-Modifikuj-Sluč s možností vynucení zamčení souboru, pokud je to nutné.

Princip ukládání dat

Data jsou ukládány v pomocí delt a v případě, že je potřeba, aby byl soubor na dvou místech v repozitáři, je na druhém místě vytvořen pouze odkaz na původní entitu, aby nedošlo ke zbytečné duplikaci dat.

Důvody nového systému

Pro vytvoření nového systému jsem se rozhodl zejména z důvodů potřeby vlastního serveru pro provoz jiných verzovacích systémů a vytvořit komplexní nástroj pro správu jak jednotlivých souborů, tak i rozsáhlých projektů.

Komplexním nástrojem je myšlen systém, který v sobě uceluje kromě samotných verzí souborů (respektive projektu) také správu dokumentace, časové rozvržení projektu společně s ukazatelem plánovaného a aktuálního postupu, možnost

Volba platformy

Vzhledem k mým několikaletým zkušenostem s vývojem webových aplikacá jsem se rozhodl pro LAMP, tedy operační systém Linux, webový server Apache, databázový systém MySQL a programovací jazyk PHP.

Operační systém Linux jsem vybral protože ho používám už několik let, jsem s ním poměrně dobře sžitý a pro vývoj webových aplikací je vhodný zejména svojí bezpečností, konfigurovatelsností a vnitřní architekturou (hiearchije adresářů je odvozena od kořene, všechny entity jsou reprezentovány jako soubor).

Apache jsem zvolil pouze proto, že na dnešních webových serverech se jedná o standard a je tedy velice pravděpodobné, že aplikace bude fungovat na většině hostingů.

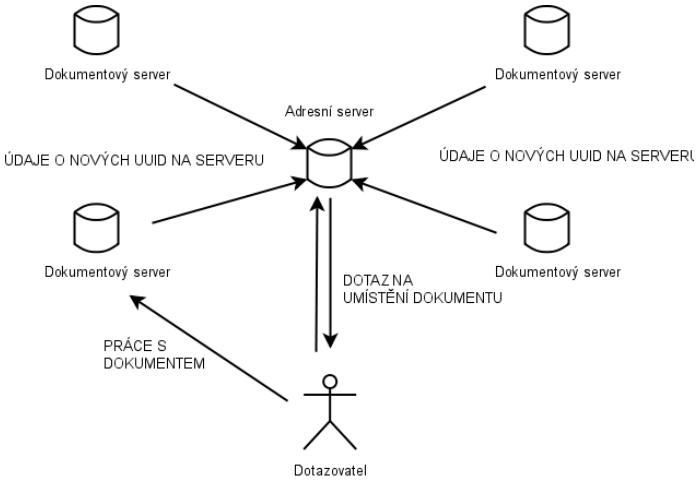
Jako databázový software jsem vybral MySQL, který je opět standardem a mám s ním bohaté zkušenosti.

S jazykem PHP pracuji již od verze 4, ve které byly poprvé implementovány objekty (velmi omezeně, použitelnou implementaci přinesla až verze 5) a vzhledem k tomu, že verze 5.3 podporuje anonymní funkce, jmenné prostory a další možnosti, není důvod proč se použití tohoto jazyka bránit. Jako framework použiji Zend, který vyvíjí přímo vývojáři jazyka a tím

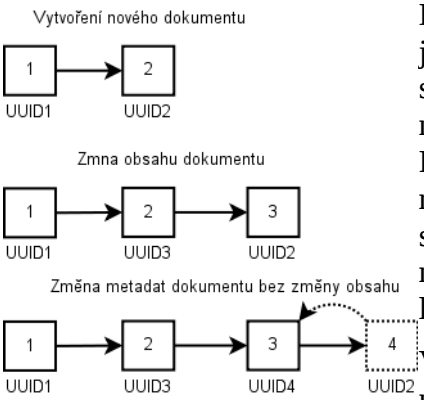
pádem je to jeden z nejschopnějších frameworků (poslední dobou mu ovšem začíná konkurovat české Nette).

Návrh systému

Ukládání dat



Vzhledem k pravděpodobné potřebě škálovatelnosti systému (rozložení úložiště na více serverů), nepoužívá program standardně používaný celočíselný inkrementovaný klíč, ale jednoznačné identifikátory dokumentu, označované jako UUID (univerzální unikátní identifikátor), založený na kombinaci UUID v.4 a UUID v.5 dle RFC 4122. Tato strategie umožňuje v případě škálování vytvořet adresní server obsahující mapu rozložení dokumentů na serverech.



Do databáze jsou ukládány pouze metadata o dokumentu, jako je jméno, přístupová práva, vlastník a vlastní obsah souboru je ukládán do adresáře na disku. Při vytvoření nového dokumentu jsou uloženy do databáze dva záznamy. První reprezentuje původní stav dokumentu a druhý reprezentuje nejauktuálnější stav. Při dalších commitech jsou staré verze vkládány mezi tyto dva dokumenty. Díky tomu má původní a aktuální verze dokumentu stále stejné UUID a lze se na něj bezproblémů odkazovat.

V případě že je commitován dokument, který se obsahově nezměnil (ale bylo mu změněno například jméno, práva), vytvoří se místo nového obsahu pouze symbolický odkaz na původní verzi dokumentu (toto řešní má smysl zejména pro velké soubory, například pokud spravujeme projekt z CAD systému).