Obsah

Verzování aplikací

Verzování je uchovávání historie veškerých změn, v závislosti na po povaze a přístupu můžeme zaznamenávat verze, jak v informacích, tak samotných datech. Nejčastěji se setkáváme s verzováním zdrojových kódu softwaru během vývoje aplikací.

Proč vlastně verzovat

Na vlastní verzování můžeme uplatnit dva pohledy. První pohled je mezi vydavatelem aplikace a jeho uživatelem. V rámci verze vydavatel uživateli říká, jaké novinky jsou v rámci změny dostupné, co už naopak aplikace nepodporuje nebo popisuje změnu chování aplikace. Verzování aplikace ale nekončí definicí verzí samotné. Stejně jako je u zálohování podstatná také obnova ze zálohy, tak je nedílnou součástí definice nové verze také umění vrátit se k verzi předchozí.

Druhý pohled je zevnitř programátorského týmu. V tomto případě nám verzování umožňuje spoluprácí většího množství programátorů na jednom softwarovém produktu, příp. sdílení částí (package, plugging) jednotlivých subsystémů mezi různými programy. V takovém případě/prostředí je nutné hlídat změny a řešit případné kolize v rámci těchto změn.

Pro řešení verzování z prvního pohledu je potřeba nají společnou řeč - tedy stanovit nějaká syntaktická pravidla pro definici verzí. Řešení problému verzování z pohledu programátorského týmu již tak triviální není a je lepší využít některého z verzovacích systémů.

Jak verzovat - major.minor.patch-{pre-release}

Obecně platí, že číslo verze je platné od chvíle, kdy je oficiálně vydána očíslovaná verze aplikace. Od této chvíle se aplikace nesmí měnit. Jakákoli změna v aplikaci musí nutné znamenat i změnu čísla verze.

Proto aby číslování jednotlivých verzi rozuměly všechny strany aplikace je nutné definovat nějakou syntaxi. Tuto syntaxi může definovat vydavatel aplikace dle libosti, musí však tuto definici uvést v nějaké dokumentaci.

Další možností je použít obecně definovanou syntaxi pro verzování - major.minor.patch-{pre-release} - např. 2.1.0-beta. Úrovně major, minor, patch jsou reprezentovány číselnou hodnotou a zvyšují se vždy o hodnotu 1. Změna vyšší úrovně (vlevo), vždy nuluje hodnotu na nižších úrovních (směrem doprava).

- Verze na úrovni *major* se mění v případě, že změny ke kterým ve verzi dochází nejsou zpětně kompatibilní s předchozí verzí nebo ostatními API. Případně se může jednat o kompletní přepsání aplikace do jiného jazyka nebo databázové struktury. Pokud je v části major uvedena 0, znamená to, že aplikace je stále ve vývoji a dle autora stabilní, přestože se zdá plně funkční.
- Na úrovni *minor* se hodnota mění v případě, že se jedná o změnu zpětně kompatibilní. Tuto úroveň zvedáme při doplnění nových vlastností a funkcí aplikace nebo v případě, že množství chyb předchozí verze přerostlo nějakou rozumnou mez a nevystačíme si se zvednutím hodnoty na úrovni patch.
- Změna hodnoty na úrovni *patch* znamená opravy chyb bez změny zpětné kompatibility. Je to čistě oprava chyby nežádoucího chování aplikace.
- Pro úroveň *pre-release* používáme většinou písmena řecké abecedy:
    - alfa - aplikace je stále ve vývoji, ale je již dostupná pro testery, většinou interní.
    - beta - aplikace je stále ve vývoji, testování ještě nebylo ukončeno, ale je již dostupná běžným uživatelům - většinou zdarma. Aplikace může být nestabilní, ale měla by být

již plně funkční. Pro vývojáře je přínos v tom, že danou verzi otestuje větší skupina uživatelů. Přenos chyb často bývá automatický, na pozadí. Pro koncové uživatele tato verze přináší první seznámení s novinkami. Skupina uživatelů s možností používání beta verzí bývá často vydavatelem software omezena.

- gamma - málo využívaná; většinou v případě, že si uživatel nevystačí s dvěma testovacími verzemi.
- U úrovně pre-release se také můžete setkat s anglickými výrazy a zkratkami - stable, pre (prelease), RC (release candidate).
- někdy bývá patch doplněn i číslem.

Změny ve verzích

Očíslování verze a vydání aplikace s daným číslem verze však není vše. Celá ta věc se vlastně dělá kvůli tomu, abychom mohli zveřejnit, co se v dané verzi změnilo. Popis změn - nové, změněné nebo zrušené funkcionality, je potřeba zanést do dokumentace aplikace a tuto dokumentaci zveřejnit současně s oficiálním vydáním dané verze aplikace. Popis změn může nebo spíš by měl být součástí nápovědy aplikace. Často také bývá součástí webových stránek aplikace, příp. papírové či elektronické dokumentace. V tomto případě platí, že je nutné udržet tento popis identický na všech používaných místech. V případě velkých systémů pro konkrétního zákazníka bývá součástí akceptace vydané verze také nějaký přehled testování dané verze - seznam chyb nebo problémů z používaného bugtrackeru.

Verzování bez verzovacího systému

Tato varianta verzování je většinou využívána při tzv. individuální vývoji aplikace. Jde o vývoj aplikace jedním vývojářem nad jedním zdrojovým kódem. Tím pádem je vyloučena kolize nad tímto kódem. I přesto se doporučuje vést nějaké verze tohoto kódu alespoň v rámci adresářové struktury souborového systému. Jednotlivé adresáře pak můžeme odlišit pomocí čísla verze v dohodnuté syntaxi. Z praxe je dobré jednotlivé verze zálohovat a na disku mít alespoň 3 verze - aktuální, předchozí a nově připravovanou verzi. Zbylé verze stačí mít na nějakém zálohovacím zařízení.

Tuto variantu lze také využít pro týmový vývoj aplikace nad kódem uloženým na nějakém sdíleném úložišti. V takovém případě je však nutné použít jeden z modelů pro správu zdrojových kódů - zamykání souborů nebo slučování verzí.

Zamykání souborů

Soubory na společném úložišti jsou dostupné všem vývojářům v rámci programátorského týmu. Jakmile daný soubor některý z vývojářů zamkne, např. zápisem změn, pak všichni ostatní mají tento soubor dostupný pouze pro čtení. Výhodou tohoto přístupu je, že nedochází ke konfliktů při slučování souboru. Nevýhoda spočívá v tom, že pokud vývojář drží daný soubor zamčený příliš dlouho, pak znemožní ostatním vyvíjet aplikaci. Často pak dojde k tomu, že si ostatní vývojáři uloží soubor lokálně a přepíší jej až ve chvíli, kdy jej první vývojář uvolní. Často pak dochází ke ztrátě změn uložených prvním vývojářem.

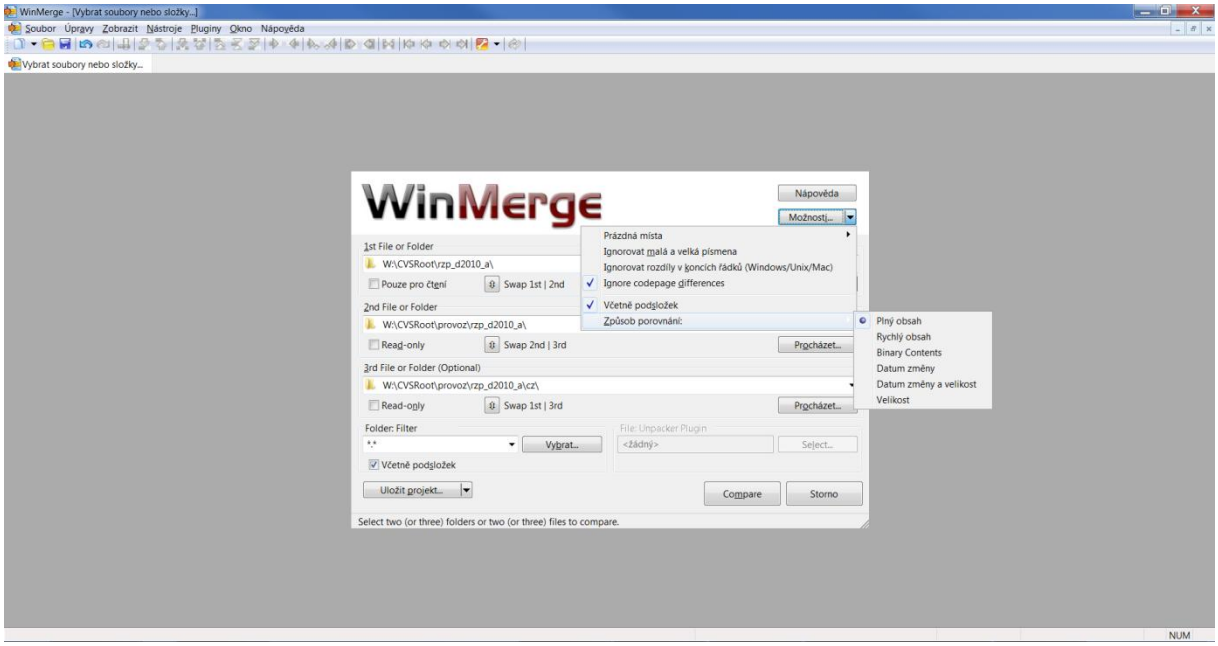
Slučování verzí

Aby nedocházelo ke ztrátě změn, je nutné sloučit soubor z centrálního úložiště s lokální kopií. V rámci verzovacích systémů máme k dispozici různé prostředky pro porovnání dvou a více souborů a také pro jejich sloučení s vyznačením případných změn. Pokud nemáme k dispozici verzovací systém, pak musíme daný soubor či soubory sloučit sami. Slučování můžeme provést okometricky - kopírováním změn mezi jednotlivými soubory - což bývá často zdrojem chyb. Nebo můžeme použít některý z programů k tomu určených. Již dříve jsme si ukázali porovnání a slučování souborů v

textovém editoru PSPad. Nevýhodou tohoto programu je přístup po jednom souboru. Pokud bychom chtěli srovnat celé adresářové struktury, můžeme použít např. program Winmerge.

Srovnání verzí - WinMerge

Tento program umožňuje srovnávat soubory, ale i celé adresáře. Mezi sebou dokáže porovnat až tři verze současně.

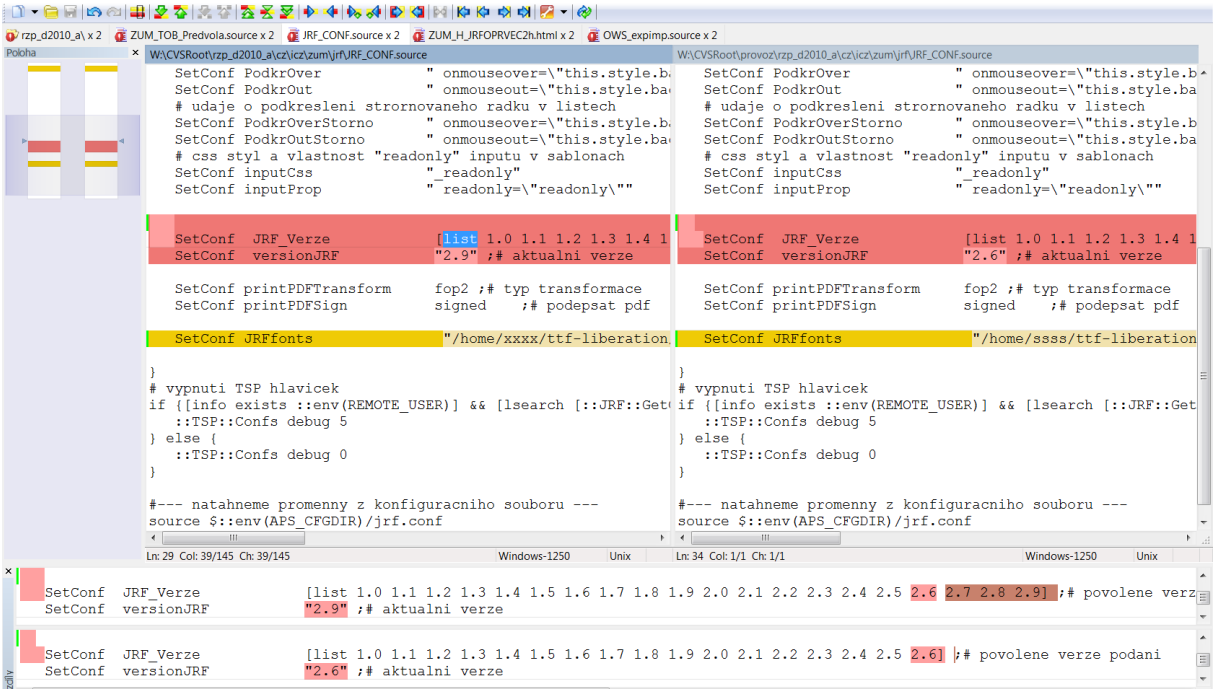


Již při výběru souborů či adresářů je možno ovlivnit jejich způsob porovnání. V případě adresářů je možné rovnou srovnat i jejich podadresáře (podsložky).

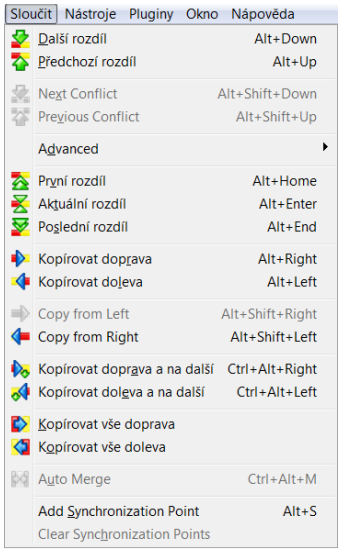
Pokud porovnáváme adresáře, pak výsledkem porovnáním mohou být 4 možnosti - soubory, které se nevyskytují v druhém adresáři (pouze vlevo). Soubory, které se nevyskytují v pravém adresáři (pouze vpravo). Oboje označeno šedě. Soubory, které jsou shodné jsou označeny bíle, a rozdílné soubor, které jsou označeny žlutě. Výsledek je vždy zobrazen i textem a patřičnou ikonou.

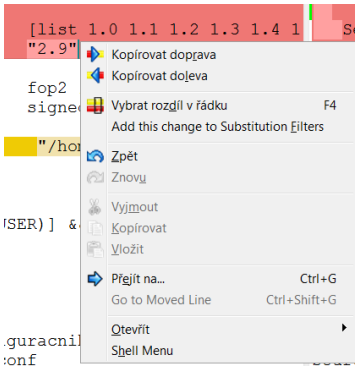
ZPUSKPS04_proc....	cz\icz\zum\zpu	Pouze vlevo: W:\CVSRoot\rzp_d2010_a\cz\icz\zum\zpu	* 17.1.2019 10:05:43		source
ZPUSKPS04_sql.s...	cz\icz\zum\zpu	Pouze vlevo: W:\CVSRoot\rzp_d2010_a\cz\icz\zum\zpu	* 8.2.2019 13:18:37		source
ZUM_H_ZUMDO...	cz\icz\zum	Pouze vlevo: W:\CVSRoot\rzp_d2010_a\cz\icz\zum	* 13.11.2019 17:10:04		html
ZUM_H_ZUMDO...	cz\icz\zum	Pouze vlevo: W:\CVSRoot\rzp_d2010_a\cz\icz\zum	* 13.11.2019 17:10:04		html
ZUM_REJ_proc2.s...		Pouze vlevo: W:\CVSRoot\rzp_d2010_a	* 25.11.2019 8:58:15		source
Tag	cz\CVS	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\CVS		* 4.5.2018 11:12:23	
Tag	cz\icz\ath\CVS	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\icz\ath\CVS		* 4.5.2018 11:08:06	
Tag	cz\icz\ath\lang\CVS	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\icz\ath\lan...		* 4.5.2018 11:08:06	
Tag	cz\icz\ath\lang\cz\...	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\icz\ath\lan...		* 4.5.2018 11:08:06	
Tag	cz\icz\ath\sql\CVS	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\icz\ath\sql\...		* 4.5.2018 11:08:06	
Tag	cz\icz\ath\sql\ifx\C...	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\icz\ath\sql\...		* 4.5.2018 11:08:06	
Tag	cz\icz\CVS	Pouze vpravo: W:\CVSRoot\provoz\rzp_d2010_a\cz\icz\CVS		* 4.5.2018 11:12:23	
TSP_H_tests.cgi	cz\sasprg\Fcelib	Textové soubory jsou rozdílné	* 10.2.2006 17:22:42	10.2.2006 16:22:42	cgi
TSP_H_tests.html	cz\sasprg\Fcelib	Textové soubory jsou rozdílné	* 11.5.2005 11:30:00	11.5.2005 10:30:00	html
TSP_H_tests.tsp	cz\sasprg\Fcelib	Textové soubory jsou rozdílné	* 11.5.2005 11:30:00	11.5.2005 10:30:00	tsp
VTF_proc.source	cz\sasprg\Fcelib	Textové soubory jsou rozdílné	* 4.9.2015 14:33:42	28.8.2015 0:58:27	source
VTF_proc2.source	cz\sasprg\Fcelib	Textové soubory jsou rozdílné	* 9.6.2005 16:47:21	9.6.2005 15:47:21	source
XSLTrans.source	cz\sasprg\Fcelib	Textové soubory jsou rozdílné	* 6.5.2019 11:44:19	3.2.2014 9:18:49	source
cz.tar.gz		Binární soubory jsou rozdílné	* 25.1.2021 8:52:49	21.5.2018 8:07:49	gz
Entries	cz\CVS	Textové soubory jsou shodné	8.4.2015 13:48:02	* 4.5.2018 11:08:04	
Entries.Static	cz\CVS	Textové soubory jsou shodné	8.4.2015 13:48:27	* 4.5.2018 11:12:23	Static
Repository	cz\CVS	Textové soubory jsou shodné	8.4.2015 13:48:02	* 4.5.2018 11:08:04	
Root	cz\CVS	Textové soubory jsou shodné	8.4.2015 13:48:02	* 4.5.2018 11:08:04	
Repository	cz\icz\ath\CVS	Textové soubory jsou shodné	8.4.2015 13:48:02	* 4.5.2018 11:08:04	
Root	cz\icz\ath\CVS	Textové soubory jsou shodné	8.4.2015 13:48:02	* 4.5.2018 11:08:04	

Vybrané rozdílné soubory se nám otevřou na jednotlivých záložkách a rozdíly mezi soubory jsou zvýrazněny opět žlutou barvou.



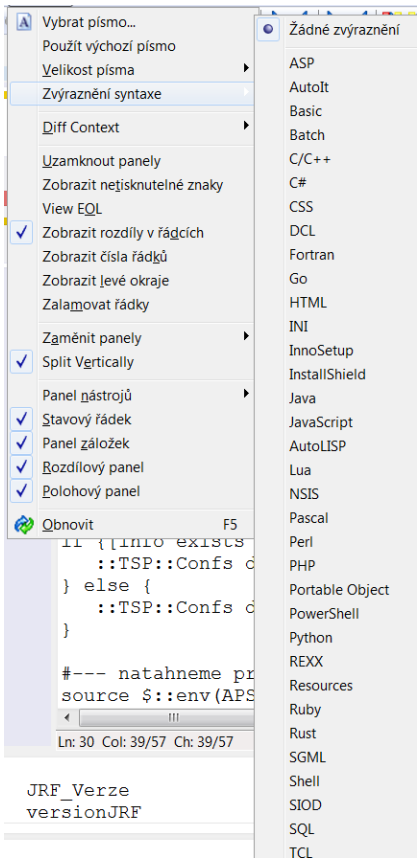
Pomocí ikon v nástrojové liště se může pohybovat po jednotlivých změnách, které jsou navíc zvýrazněny ve sloupci vlevo. Aktivní změna je zvýrazněna červeně a v pracovní okně dole jsou zobrazeny změny na konkrétních řádcích. Další ikony nám pak umožní přenášet jednotlivé změny, příp. všechny změny oběma směry. Příp. lze akce vyvolat z menu Sloučit nebo kontextové menu.



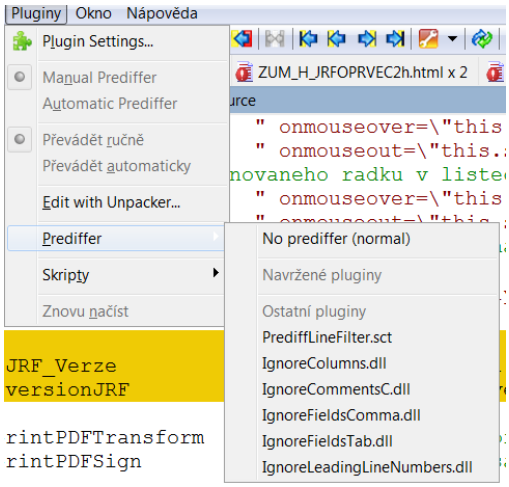


Program také umožňuje uložit protokol porovnání verzí do html souboru, příp. export protokolu srovnání adresářů do formátu txt či csv pro pozdější načtení do databáze nebo programu typu MS Excel apod.

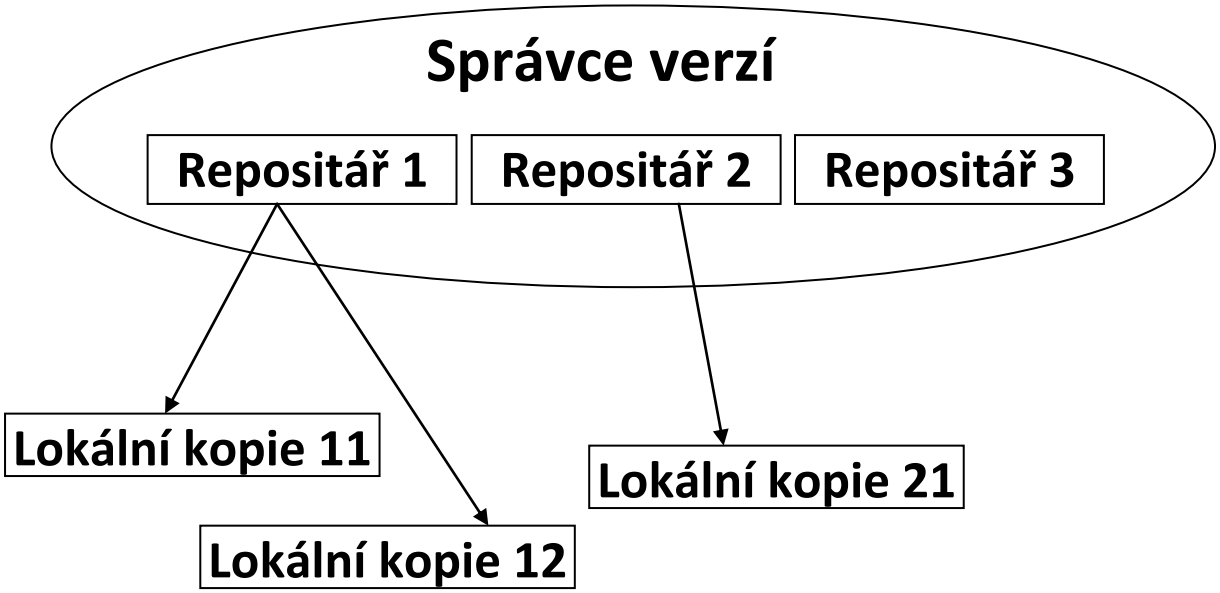
Uživatelsky je také možno ovlivnit zobrazení jednotlivých změn, příp. nastavit syntaxi jazyka v porovnávaných souborech.



A jak jsme zvyklí již z vývojových prostředí lze program doplnit o některý z předdefinovaných skriptů pluginů nebo si definovat vlastní.



Verzovací systémy



Verzovací je v podstatě správce verzí, který má na starosti automatické číslování verzí. Poskytuje nějakou sadu nástrojů (příkazů) pro přístup k jednotlivým repositářům. To jsou místa, ve kterých jsou uloženy verzované soubory a adresáře. Přístup uživatelů k těmto repositářům je umožněn přes lokální kopie těchto repositářů, resp. verze souborů uložených v těchto repositářích. Lokální kopií může být lokální repositář nebo pracovní adresář. Soubory v pracovním adresáři pak jednotliví uživatelů mohou modifikovat a poté jej promítnou zpět do repositáře, se kterým je lokální kopie spojena.

Základní nástroje/příkazy pro komunikaci se správcem verzí

checkout

Příkaz checkout slouží k získání lokální kopie souborů z repositáře. V této lokální kopii pak dochází ke změnám jednotlivých souborů. Tato změna nemá vliv na repositář. Správce verzí při vykonání tohoto příkazu často do lokální kopie vytvoří adresář s pracovními informacemi k jednotlivým souborům, informacemi o vlastním repositáři i o správci verzí a připojení k němu.

add

Tento příkaz slouží k vlastnímu přidání souboru k aktuálnímu repositáři. Dokud jej neprovedeme, tak se jedná pouze o běžný lokální soubor, o kterém repositář nevím a jehož se žádné příkazy týkat nebudou.

commit

Provedením tohoto příkazů požádáme o přijetí modifikací z lokální kopie do repositáře. Každý commit je **nutné** řádně **okomentovat**. Toto se často z pohodlnosti ignoruje a dříve nebo později se to uživateli po zásluze "vrátí". Pokud je změna bezkonfliktní, pak správce verzí vytvoří nejbližší vyšší verzi souboru a změnu přijme. Pokud změna není bezkonfliktní, pak je nutné tento konflikt vyřešit.

update

Příkazem update si uživatel stáhne aktuální verzi souboru z repositáře do své lokální kopie. Pokud v lokálním adresáři nemá žádnou modifikaci, pak dojde pouze k prostému stažení nové verze souboru. V případě, že se pokoušíme stáhnout novou verzi souboru k vlastnímu modifikovanému souboru a tato změna je bezkonfliktní, pak dojde k automatickému sloučení změn (merge) v souboru. Pokud není změna bezkonfliktní, pak je nutné tento konflikt vyřešit.

diff

K řešení konfliktů slouží příkaz diff, jehož prostřednictvím dojde ke zvýraznění změn mezi lokální kopií souborů a souborem v repositáři. Konflikt nejčastěji vzniká modifikací stejného místa v souborů.

Obecně platí, čím méně konfliktů, tím rychlejší práce. Proto se snažíme konfliktům předcházet např. rozčleněním aplikace na moduly, ve kterých jednotliví uživatelů pracují samostatně.

tag

V rámci vývoje je dobré v určitém okamžiku označit určité místo v repositáři značkou tagem. Často používáme stejnou značku pro více souborů najednou a v podstatě tak vytváříme verzi aplikace, jak jsme si definovali v předchozí kapitole např. release_1.1.0. Aktuální verze používá značku HEAD.

branch

Pokud chceme vyvíjet paralelně, resp. v týmu, pak se neobjedeme bez takzvaného větvení vývoj. Vznik jakékoli nové větve vývoje je potřeba řádně zdokumentovat V čase je pak nutné jednotlivé verze uzavírat (vracet do hlavní vývojové větve). Obecně je dobré nemít příliš otevřených vývojových větví.

V rámci vývoje rozlišujeme větve:

- STABLE - stabilní vývojová větev
- CURRENT, TRUNK - hlavní vývojová větev
- DEVELOP - větev pro vývoj konkrétního problému. Těchto větví bývá obecně více.