Otázka 16 - Y36SI3

Zadání

Disciplinované přístupy ke změnám software (SCM). Nástroje pro správu a verzování zdrojového kódu. Řešení konfliktů v nástrojích pro správu zdrojového kódu. Slučování změn (operace merge). Stavový diagram požadavku. Integrace nástroje pro správu zdrojového kódu s nástrojem pro správu požadavků. (Y36SI3)

Disciplinované přístupy ke změnám software (SCM)

SCM = Source Code Management software, základní typologie rozdělení je podle počtu a umístění základního úložiště kódu(=repository) na:

- **Centralizované** - příklad: SVN(Subversion), CVS
- **Distribuované** - příklad: Git, Mercurial

Centralizované

můžeme charakterizovat tak, že máme jednu **ústřední repository**, do které vývojáři nahrávají (commitují) změny kódu a udržují verze vyvíjeného software. Vývojář tak musí mít **stále k dispozici připojení** k serveru, kde je daná repository (obvykle tedy potřebuje mít připojení k internetu).

Výhody:

po **každém** commitu vývojáře je nová verze souborů k dispozici ihned celému týmu.

Nevýhody:

Vývojář **nemůže commitovat** pokud nemá k dispozici připojení k repository. Řekněme tedy, že vývojář odjíždí v pátek na chatu a v neděli zjistí, že během soboty udělal fatální chybu. Vzhledem k tomu, že o víkendu byl na chatě a nemohl tedy commitovat, nemůže se vrátit k původní verzi.

Konkrétní řešení:

- CVS - Concurrent Version System - již starší systém
- SVN - Subversion
- Rational ClearCase
- Microsoft Visual Source Safe
- Microsoft TFS

Distribuované

nelze tak snadno charakterizovat - repository existuje v rámci týmu vývojářů **vícero** (např. každý vývojář má na svém PC vlastní repository) a výsledný produkt vzniká často různým spojováním verzí z různých repository nebo je vytvořen strom repository a změny se do vyšších větvích promítají postupně. Vývojář tedy commituje změny do své vlastní repository a do centrální je pak operací **push** pošle až po delší době. Pokud chce vývojář získat současnou verzi z centrální repository či od jiného kolegy, volá operaci **pull**. Je také obvyklé, že před nahráním verzí do centrální repository si vývojáři sdílejí své repository mezi sebou.

Další možný scénář je, že existuje několik centrálních repository, které vlastní někteří členové týmu.

Výhody:

- Při většině operací **není využívána síť** - vývojář může commitovat kdykoli, protože repository má vždy přístupnou - viz. předchozí příklad s chatou, vývojář kdyby během svého víkendnového pobytu commitoval do své repository na svém PC, pak by se k sobotní chybě mohl lehko vrátit a napravit.
- Zálohování - jelikož repository jsou na více počítačích (obvykle všech vývojářů), pravděpodobnost **ztráty dat** se tak výrazně **snižuje**.
- **Rychlost** - vývojář commituje změny do své lokální repository, obvykle se tedy nepřipojuje ke vzdálenému serveru, což šetří čas a taková operace je mnohem rychlejší než operace commit u centralizovaného řešení.
- Možnost sdílení jednotlivých repository mezi sebou

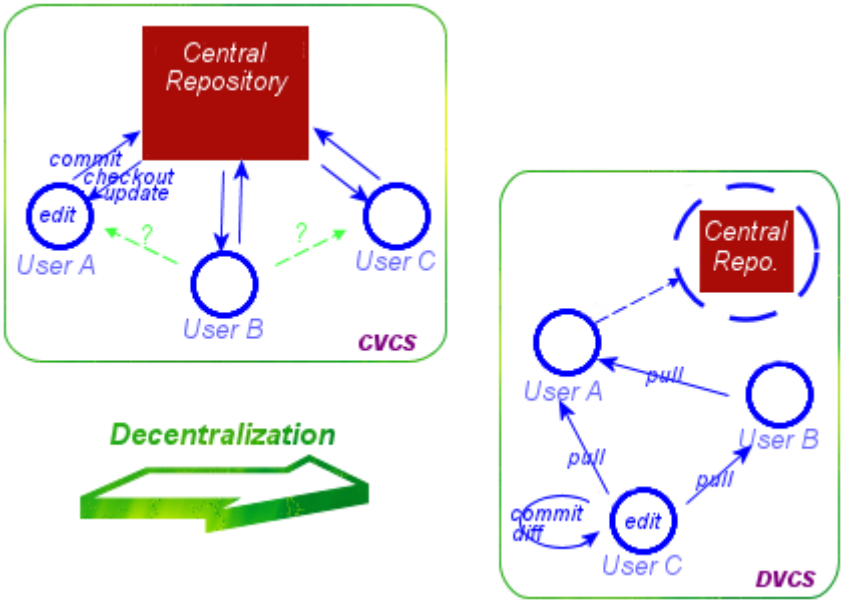
Nevýhody:

Více repository, více současně existujících verzí, **ztráta** přehlednosti. Větší pravděpodobnost vzniku **konfliktů** (vývojář svou repository nahrává ostatním obvykle až po nějaké době, změn je tedy více). Nicméně výhody obvykle převažují.

Konkrétní řešení:

- Git
- Mercurial
- Arch
- Darcs

Centralizované vs. distribuované



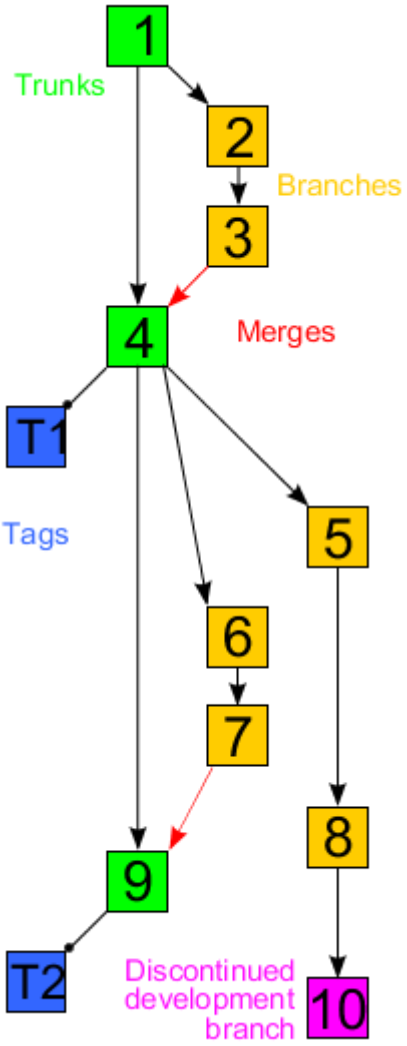
Zatímco u centralizovaného řešení vývojaří komunikují **pouze** s centrální repository, kde je aktuální verze, u distribuovaného je repository více, které mohou komunikovat i mezi sebou, a do centrální repository se nahrává obvykle již mezi vývojáři prodiskutovaná verze.

Názvosloví pro SCM

- *checkout* - získání celé aktuální verze z repository
- *update* - získání změn z repository, změny jsou jen mezi lokální kopií a aktuální v repository
- *commit* - nahrání změn do repository
- *branch* - zkopírování vývojové či jiné větve do nové větve

- *tag* - označení aktuálního stavu větve nějakým unikátním jménem
- *merge* - spojení dvou souborů (větví) - nahrání změn v jedné větvi do druhé větve - viz. níže
- *blame(annotate)* - výpis řádků daného souboru, kdy na začátku každého řádku je vidět, kdo ho naposledy měnil. Blame nám tedy říká, kdo který řádek napsal a můžeme tak obvinit (blame) daného autora.
- *Pull* - u distribuovaných systémů stáhnout data z nějaké repository do své repository
- *Push* - u distribuovaných systémů poslat změny ze své repository do jiné (např. centrální) repository

Řešení konfliktů v nástrojích pro správu zdrojového kódu. Slučování změn - merge.

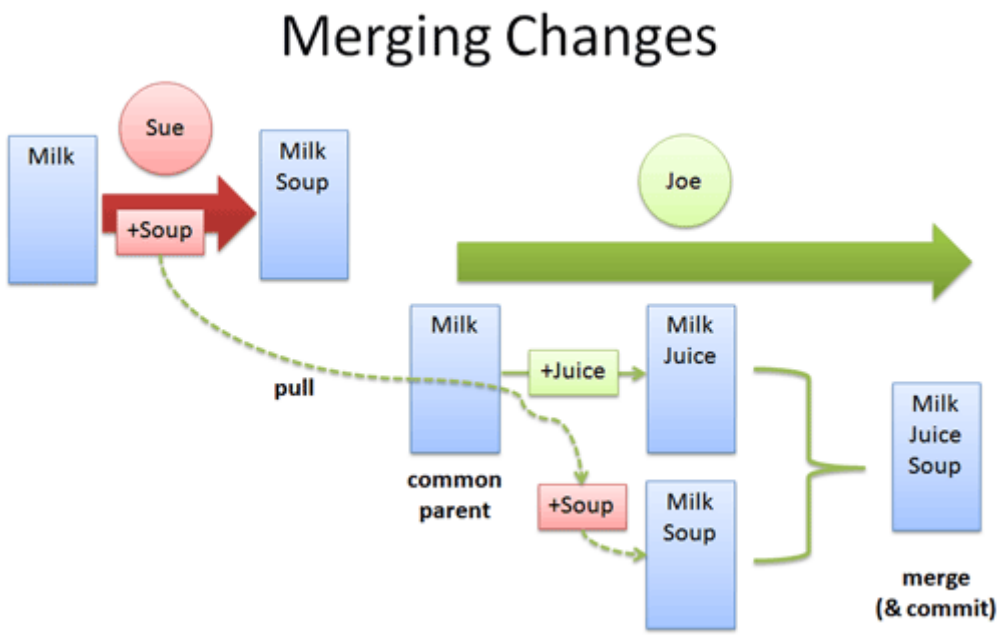


U centralizvaných systémů

Když jeden vývojář commitne svou novou verzi souboru a druhý vývojář posléze commituje jinou svou verzi stejného souboru, konflikt nastane ve chvíli, kdy oba tito vývojáři dělali změny na stejných řádcích souboru. Vývojář, který commitoval jako druhý pak musí tento konflikt vyřešit. Obecně lze říci, že tento konflikt se vyřeší operací **merge**. Vývojář tak sloučí tyto dva soubory (verze) do jednoho, a to tak, že musí vybrat, které řádky z které verze budou ve verzi konečné - jedná se tedy o sloučení dvou souborů, přičemž řádky mohou pocházet z obou - od toho název sloučení (merge).

U SVN (Subversion) vypadá operace merge takto: Druhý vývojář dělá před svým commitem nejdříve update, aby se ujistil, že nedochází ke konfliktu, pokud ano, vytvoří se *.mine, *.r93, *.r94 (čísla u přípon se samozřejmě mění). Máte tedy dvě možnosti: využít operaci merge a soubory *.r93 a *.r94 sloučit do *.mine, nebo verzi *.mine editovat ručeně a vybrat které řádky z jakého souboru ponechat. Následně se provede příkaz Resolved, čímž se SVN ujistí, že konflikty byly vyřešeny a následně commit do centrálního repository.

U distribuovaných systémů



Jednotlivé soubory obvykle existují ve více repository současně, a tak při úpravách stejného souboru více vývojáři vznikají tzv. větve (branches). Z obrázku vidíme, že Sue a Joe upravují stejný soubor, který obsahuje *Milk*. Sue přidá do tohoto soubor *Soup* a Joe *Juice*. Joe zavolá operaci **pull** a stáhne si tento soubor od Sue, následně zavolá operaci **merge** a tím sloučí tyto dva soubory. Nakonec provede **commit** čímž potvrdí sloučení změn. Nyní stačí, aby Sue zavolala **pull** a **update**, čímž si načte změny, které provedl Joe.

Issue tracking a bug tracking software = nástroje pro správu požadavků

Požadavek je v tomto případě požadavek na práci člena teamu, na doplnění/opravení funkčnosti systému. Dále se termín *požadavek* používá v souvislosti s nahlášením problému, úkolu, bugu apod. Tato sekce se nezabývá tzv. požadavky (requirements) na systém, které se obvykle definují při analýze projektu - o tom je otázka č. 12. Systém pro správu projektu by měl mít systém pro správu požadavků. Anglicky ticket, issue, task.

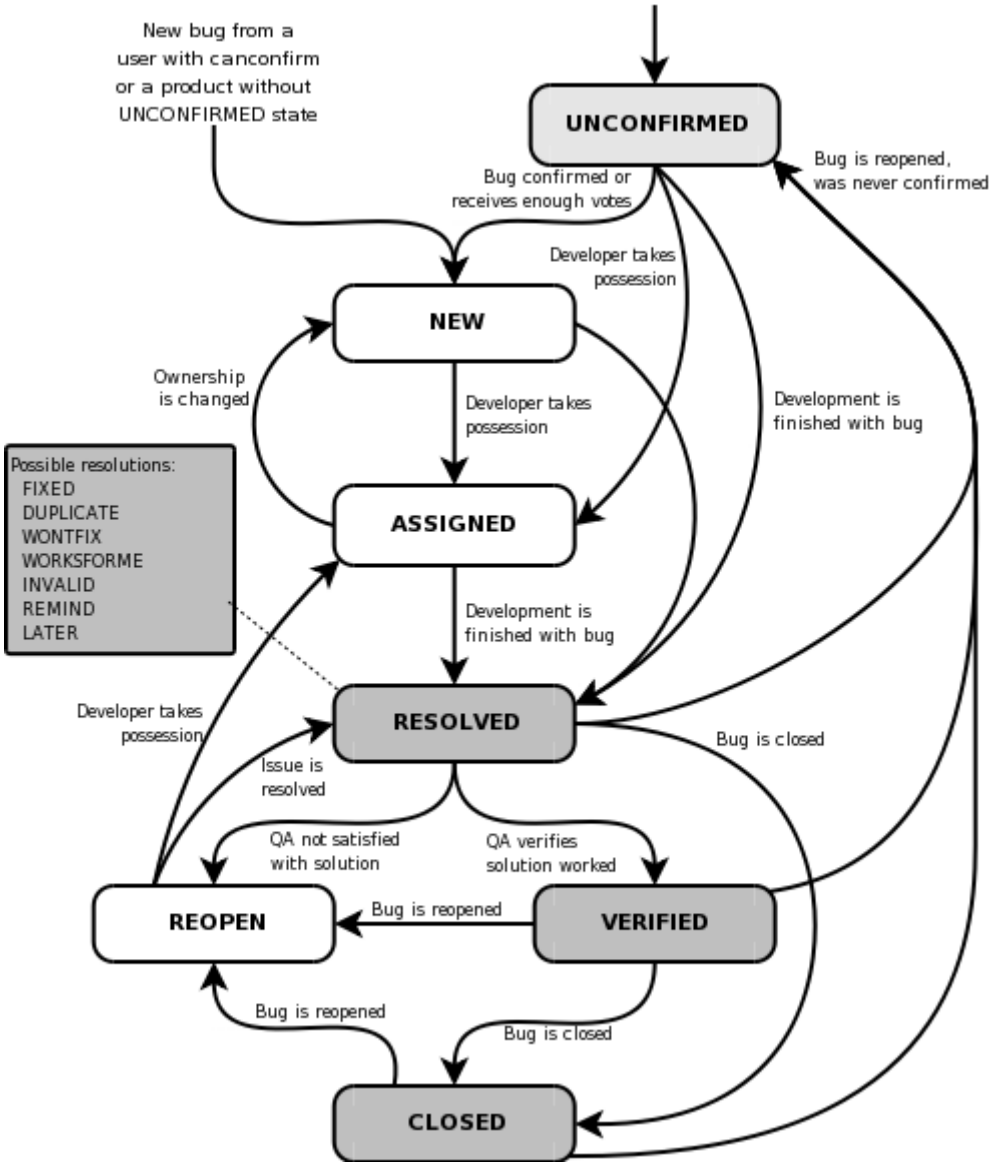
Zjišťování požadavků probíhá:

- od zákazníka
- při zadávání práce členům týmu
- při testování

Uspořádání požadavků:

- Požadavky mají různý typ, např.: chyba, vylepšení, úkol
- Požadavky mají různou prioritu, např.: trivial, minor, major, critical, blocker
- Požadavky mohou náležet k různým komponentám jako např.: database, user interface, organization, presentation, kernel, …
- Požadavky mají různé stavy jako např.: new, reopened, assigned, invalid, duplicate, fixed
- Požadavky se také přiřazují jednotlivým lidem, k milestonům nebo k verzím programu

Životní cyklus bugu/požadavku v Bugzille:



Řešení

- Trac
- Bugzilla
- BugTracker.NET
- IBM Rational ClearQuest

Integrace nástroje pro správu zdrojového kódu s nástrojem pro správu požadavků

Mezi nástroje, které mají intergrovaný systém pro správu zdrojového kódu a pro správu požadavků, patří např. TRAC. Na jeho příkladě se dá snadno ukázat princip a výhody integrovaného systému. Trac má mimojiné v sobě zabudovanou Wiki, Issue tracking system (pomocí tickets) a SCM (obvykle Subversion, ale poskytuje interface i k jiným).

Mezi těmito 3 části je možné vytvářet **reference a odkazy**. To znamená, že můžeme využít tzv. syntaxi **TracLinks** a použít ji ve zprávách při commitu, na wiki a ticketech a milestonech a to tím, že napíšeme např:

- **#1 nebo ticket:1** - odkaz na ticket s číslem 1
- **wiki:CamelCase** - odkaz na stránky Wiki
- **source:/trunk/COPYING@200#L25** - odkaz na 25. řádku v souboru COPYING ve verzi 200

Mimoto je možné využít Wiki syntaxi a formátovat tak náš text.

Mezi produkty, které mají integrované nástroje pro SCM a správu požadavků patří:

- Trac
- Assembla
- Polarion Trac and Wiki

Slovníček pojmů

SCM = Source Code Management - správa zdrojového kódu
SVN = Subversion
Issue tracking = správa požadavků
Bug tracking = správa chyb (bugů)
Ticket = požadavek, úkol **Milestone =** určitá fáze projektu s jasným deadlinem