Správa verzí, možnosti verzování, typické situace při správě verzí (větvení, značkování), nástroje pro správu verzí, vazba na správu změn.

Wednesday, May 29, 2013 4:57 PM

Správa verzí

správa verzí je součást úlohy konfiguračního managementu – identifikace konfigurace

účelem je udržení přehledu o podobách prvků konfigurace

- verze popisuje jednu konkrétní podobu
- v úložišti jsou skladovány všechny verze

druhy verzí

- evoluční = revize (př. Word 6.0)
- alternativní podoba = varianta (př. Word pro Macintosh)

určení konkrétní verze

- **verzování podle stavu** (verze prvku) – identifikují se pouze prvky
- **verzování podle změn** (identifikace změny prvku) – identifikují se také změny prvků, výsledná verze prvku vznikne aplikací změn

Možnosti verzování

granularita

- Jednotlivé prvky (verzování komponent)
  - Konfigurace nemá verzi
- Celé konfigurace (úplné verzování)
  - Verze konfigurace indikuje verze prvků
- Verze produktu

popis verze

- extenzionální verzování: každá verze má jednoznačné ID
  - major.minor + build schéma- např. 6.0.2800.1106 (MSIE 6)
  - kódové jméno: One Tree Hill (= Firefox 0.9)
  - marketingový: Windows 95
- intenzionální verzování: verze je popsána souborem atributů
  - např. OS=DOS and UmiPostscript = YES
  - C preprocesor umožňuje intenzionální stavové verzování - např. chceme variantu foo.c pro případ OS=DOS and UmiPostscript=YES

Možnosti verzování

- Rychlý přístup k jakékoli historické nebo alternativní verzi
- Možnost vytvoření branch, tagu => částečná izolace ale s možností aplikace vývoje v trunku
- Pojmenovávání milestones
- Celý tým má přístup k aktuálnímu stavu vývoje
- Aktuální stav vývoje je jednoznačně určen
- Soukromý pracovní prostor v rámci nejnovější nebo vybrané verze
- Možnost testování lokální změny a commitu až funkční a otestované součásti
- Možnosti pro řešení konfliktů
- Některé verzovací systémy jsou inherentně zálohovací (GIT)

Informace o verzi

- Identifikátor verze (extenzionální) - klíčovým požadavkem je jedinečnost
- "major.minor.micro + build" - např. 6.0.2800.1106 (MSIE6)
- Záleží na použitém nástroji a vztahuje se na prvky konfigurace
- Marketingové jméno se pak dává celému produktu (Longhorn)
- Další meta-data prvku jsou datum/čas vytvoření, autor, stav prvku/konfigurace, předchůdci…

Prostředí pro verzování: úložiště

- Úložiště (databáze projektu, repository) = sdílený datový prostor, kde jsou uloženy všechny prvky konfigurace projektu
  - Zdrojové kódy
  - Knihovny (přeložené) a kód třetích stran
  - Konfigurační soubory, datové soubory
  - Scripty pro build, testování a instalace
  - Dokumentace, modely, prototypy
  - Odpadkový koš
- Řízený přístup (udržení konzistence)

Práce s úložištěm

- Základní operace
  - Inicializace - vytvoření úložiště, naplnění bootstrap verzí projektu
  - Check-out - kopie prvku do lokálního pracovního prostoru
  - Check in (commit) - uložení změněných prvků do úložiště
  - Zjišťování stavu - sledování změn z úložišti versus v pracovním prostoru
- Přístup k zamykání při check in/check out
  - Read-only pro všechny
  - Pesimistický: read-write kopie prvku jen pro pověřeného
  - Optimistický: read-write pro kohokoli, řešené konfliktů

Pracovní prostor

- Workspace = soukromý datový prostor, v němž je možno provádět změny prvků konfigurace, aniž by byla ovlivněna jejich oficiální podoba používaná ostatními vývojáři

Typické situace při správě verzí

Správa verzí, možnosti verzování, typické situace při správě verzí (větvení, značkování), nástroje pro správu verzí, vazba na správu změn.

- **Správa verzí**
  - Součást úlohy konfiguračního managementu – identifikace konfigurace
  - Účelem je **udržení přehledu o podobách prvků konfigurace**
  - **Verze popisuje jednu konkrétní podobu**
  - V úložišti jsou **skladovány všechny verze**
  - Popis verze
    - **Extenzionální**: major.minor+build schéma, např. 6.0.280.1106 (MSIE 6)
    - **Intenzionální**: např. OS=DOS
  - Určení konkrétní verze
    - **Verzování podle stavu** (verze prvku) – identifikace pouze prvků
    - **Verzování podle změn** (identifikace změny prvku) – id. také změn prvků <- aplikace změn
  - *Granularita verzí: verze prvku -> verze konfigurace -> verze produktu*
- **Možnosti verzování**
  - Rychlý přístup k jakékoli historické nebo alternativní verzi
  - **Možnosti vytvoření branch** => částečná izolace, ale s možností aplikace vývoje v trunku
  - **Pojmenování milestonů = tag**
  - Celý tým má přístup k aktuálnímu stavu vývoje
  - Aktuální stav vývoje je jednoznačně určen
  - Soukromý pracovní prostor v rámci nejnovější nebo vybrané verze
  - Možnosti testování lokální změny a commitu až funkční a otestované součásti
  - **Možnosti pro řešení konfliktů**
  - Některé verzovací systémy jsou inherentně záhovací (GIT)
  - **Možnost paralelní práce na stejné konfiguraci**
    - Delta diff = množ.změn prvku konfigurace mezi dvěma po sobě následujícími verzemi
    - Větvení a spojování: trunk (hlavní), branch (paralelní), merge (sloučení hlavní + paral)
- **Typické situace při správě verzí (větvení, značkování)**
  - **Trunk**: hlavní vývojová větev (kmen)
  - **Branch**: větev – „soukromý“ vývojový prostor
  - **Merge**: sloučení větve do trunku a řešení konfliktů
  - **Tag**: značkování – označení milestounů, release, apod.
  - **Kolize a konflikty** – sloučení dvou konfliktních prvků
- **Nástroje pro správu verzí**
  - Centralizované:
    - **CVS** (Concurent Versions System) – práce s celými konfig., optimista při sluč. změn)
    - **SVN** (Subversion) – založen na CVS, verzuje změny (incrementální číslování revizí)
  - Decentralizované
    - **GIT** – pro nelineární vývoj, lokální repository
    - **Mercurial**, **Bazaar**
- **Vazba na správu změn**
  - Vazba revize na ticket/change request
  - Možnost požadavků na opravu / update konkrétních verzí

Trunk: Hlavní vývojová větev

Branch: Větve – „soukromý“ vývojový prostor

Merge: Sloučení větve a do kmene a řešení konfliktů

Tag: Značkování – označování milestounů, release apod.

Kolize a konflikty – diff, sloučení dvou konfliktních commitů…

Postup vývoje a verzování

- Check out výchozí verze
- Vývoj
- Lokální testování a opravy
- Check in nové verze
- Integrační testy a opravy
- Check in nové baseline

Codeline (vývojová linie)

- Je to série podob (verzí) množiny prvků konfigurace tak, jak se mění v čase
- Má přiřazena pravidla práce s codeline (kdy a jak je možno provádět změny…)
- Vrchol codeline obsahuje nejčerstvější verzi (head)

Tag (label)

- Označení konfigurace symbolickým jménem

Baseline

- Konzistentní konfigurace tvořící stabilní základ pro produkční verzi nebo další vývoj
    - Příklad: milník "stabilní architektura", beta verze aplikace
    - Stabilní: vytvořená, otestovaná a schválená managementem
    - Změny prvků baseline jen podle schváleného postupu
    - Při problémech návrat k baseline

Paralelní práce na stejné konfiguraci

- Důvod: velké úpravy, release, spekulativní vývoj, varianty…
- Cíl: vzájemná izolace paralelních prací tak, aby ukládané změny během nich neovlivnily ostatní (oddělení paralelních vývojových linií) -> cena za to je následné řešení konfliktů

Delta, diff

- Delta = množina změn prvku konfigurace mezi dvěma po sobě následujícími verzemi
- V některých systémech jednoznačně identifikovatelná
- Changeset: delta + důvod
- Diff a patch = rozdíl mezi verzemi (text, binární), aplikace rozdílu na verzi - viz changeset

Větvení a spojování

- Kmen (trunk, master) - hlavní vývojová linie
- Větev (brach) - paralelní vývojová linie - operace vytvoření větve = branch off, split
- Spojení (merge) - sloučení změn na větvi od kmene
    - Slučuje se delta od branch off nebo posledního merge
    - Řešení konfliktů: automatizace vhodná, ale ne vždy možná
    - 2-way a 3-way merge

Distribuovaný vývoj

- Geograficky distribuovaný tým, v čase distribuovaný tým, offline práce se synchronizací, experimentální lokální úložiště
- Možnosti: centrální úložiště s privátními větvemi, distribuovaný verzovací systém

Nástroje pro správu verzí

Centralizované

- RCS: revision control systém
    - Pesimista je to
    - Pracuje s jednotlivými soubory, nepodporuje projekty
    - Historie všech změn vč. autorů
    - Ukládá rozdíly
    - Umožňuje zamykání
- CVS: current versioning systém
    - Práce s celými konfiguracemi a projekty najednou
    - Optimista – slučování změn
- SVN: subversion
    - Velmi podobný CVS (následník)
    - Verzuje celé úložiště (inkrementální číslování revizí)
    - Souborová struktura

Decentralizované

- Každý uživatel má kompletní lokální kopii repozitáře (klony)
- Lokální commity, na centrální server lze nahrát víc commitů najednou
- GIT – jádro linuxu
    - Velmi nelineární vývoj, recenzování a začleňování
    - Nelze měnit historické verze
- Mercurial – Netbeans, OpenJDK, Symbian OS
- Bazaar – Ubuntu

- **Ruční verzování**
- **Základní** - správa verzí souborů
    - Obvykle extenzionální verzování modulů
    - Centrální úložiště
    - Ukládání všech verzí v zapouzdřené úsporné formě
    - Příklad nástrojů: rcs, cvs, subversion…
- **Distribuované**
    - Více úložišť, synchronizace
    - Flexibilnější postupy

        - Příklad nástrojů: SVK, git, Mercurial
    - **Pokročilé** - integrace do CASE
        - Obvykle kombinace extenzionálního a intenzionálního verzování
        - Automatická podpora pro check in/check out prvků z repository do nástrojů
        - Příklad nástrojů: ClearCase, Adele

Co nástroj má umět

- Operace s úložištěm
- Verzování
- Podpora týmu a procesu - vzdálený přístup, konfigurovatelné zamykání a přístupová práva, automatické oznamování, spouštění scriptů při operacích, Integrace do IDE, řádkové a webové rozhraní

Rcs

- Správa verzí pro jednotlivé textové soubory
- Ukládá historii všech změn v textu souboru
    - Informace o autorovi, datu a času změn
    - Textový popis změny zadaný uživatelem
    - Další info
- Používá diff(1) pro úsporu místa - poslední revize je uložena celá, předchozí jen pomocí diff
- Funkce: zamykání souborů, symbolická jména revizí, návrat k předchozím verzím, možnost větvení a merge

CVS

- Concurrent Versioning Systém
- Práce s celými konfiguracemi (projekty) najednou
- Sdílené úložiště + soukromé pracovní prostory
- Optimistický přístup ke kontrole paralelního přístupu (zkopíruj modifikuj sluč)
- Zjišťování stavu prvků, rozdílů oproti repository
- Integrace do mnoha IDE a CASE nástrojů

Subversion

- Následník CVS
- Bez omezení předchůdce - přejmenování, verzování adresářů, atomický commit, http přístup
- Nové možnosti - binární diff, meta-data, abstraktní síťová vrstva (DAV), čisté API
- Způsob práce a příkazy velmi podobné CVS
- Identifikace verzí - globální kontinuální celočíselné identifikátory - číslují commit

Vazba na správu změn

- Vazba revize na ticket/change request
- Možnost požadavků na opravu / update konkrétních verzí (např. long-term support)

From <https://d.docs.live.net/e3534876709763a3/Dokumenty/ZCU/Statnice/Statnice.docx>