You spend more time coordinating with (and tripping over) the other team members than you do programming.

-- W.Babich

Konfigurační řízení (změny, verze, sestavení)

ASWI 2014/2015

Výchozí problém(y) a motivace

- Při vývoji produktu ve více verzích a/nebo ve více lidech je nutné zamezit zmatkům při
  - realizaci jednotlivých uživatelských požadavků
  - implementaci změnových požadavků přidávaných za pochodu
  - opravování chyb na různých verzích
  - provádění změn na jednom objektu (dokumentu, tabulce)
  - vytváření a označování spustitelné verze
  - zjišťování aktuálního stavu vývoje, nasazené verze

13. ČERVNA 2013 7:00 | LIDOVKY.CZ > BYZNYS > AUTO

Volkswagen svolá v Austrálii k opravě téměř 26 tisíc vozů

FOTO: REUTERS

CANBERRA Německý automobilový koncern Volkswagen svolá v Austrálii k opravě téměř 26 000 vozů. Důvodem jsou možné problémy s převodovkou. Kvůli podobným potížím automobilka již dříve svolala zhruba 91 000 vozů v Číně.

Zprávy o problémech se ztrátou výkonu motoru v autech Volkswagen teď vyšetřuje australská vláda. Vyšetřování souvisí s nehodou, při které v roce 2011 zahynula řidička modelu Volkswagen Golf.

Opatření v Austrálii se týká modelů Golf, Jetta, Polo, Passat a Caddy, vyrobených od června 2008 do září 2011. Podle agentury Reuters je opatření reakcí na stížnosti majitelů automobilů. Ti

Řešení

Konfigurační řízení

Co je Konfigurační management

- Proces identifikování a definování prvků systému, řízení změn těchto prvků během životního cyklu, zaznamenávání a oznamování stavu prvků a změn, a ověřování úplnosti a správnosti prvků [IEEE-Std-729-1983]
  - „jak vytvářet, sestavovat a vydávat produkt, identifikovat jeho části a verze, a sledovat změny“

- Software Configuration Management (SCM)
- řízení konfigurace, konfigurační řízení

Terminologie: Prvek konfigurace

- Prvek konfigurace = konstituující složka systému
  - různé typy: zdrojový soubor, dokument, model, knihovna, script, spustitelný soubor, testovací data, …

- Ve správě SCM

  => ví se o jeho existenci, vlastníkovi, změnách, umístění v produktu
  - atomický z hlediska identifikace, změn
  - jednoznačně identifikovatelný: např. MSW/WS/IFE/SD/01.2
    - typ prvku (dokument, zdrojový text, testovací data)
    - označení projektu
    - název prvku
    - identifikátor verze

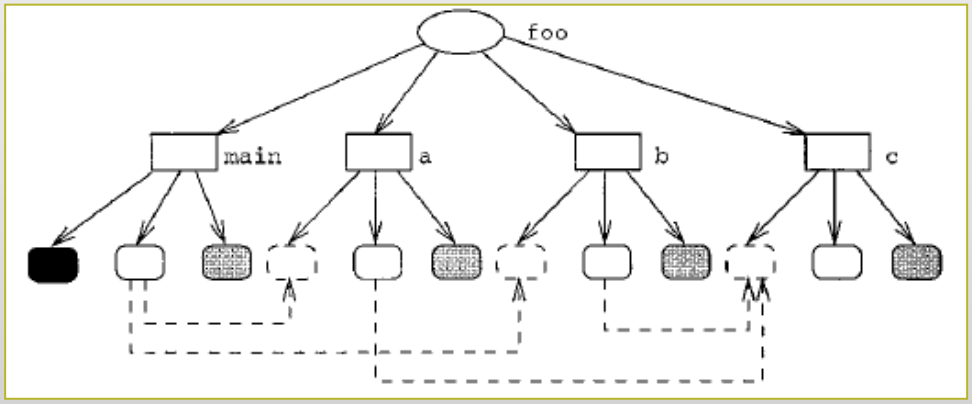
Konfigurace

- SW konfigurace = sestava prvků konfigurace reprezentující určitou podobu daného SW systému
  - příklad: „první kompilovatelná verze programu XY pro Linux“
  - v konfiguraci musí být vše, co je potřebné k jednoznačnému opakovatelnému vytvoření příslušné verze produktu
    - včetně překladačů, build scriptů, inicializačních dat, dokumentace
  - může být jednoznačně identifikovatelná

- Konzistentní konfigurace = konfigurace, jejíž prvky jsou navzájem bezrozporné
  - příklad: zdrojové soubory jdou přeložit, knihovny přilinkovat
  - těsná souvislost SCM a QA

Popis konfigurace

- Struktura produktu jako množiny prvků a jejich vzájemných vztahů
- Prvek = výsledek sw procesu
  - různá granularita a reprezentace (soubor/modul/deklarace)
- Vztahy
  - celek-část, master-dependent – určují strukturu a závislosti
  - zdrojový-odvozený – určují způsob produkce, tj. build produktu

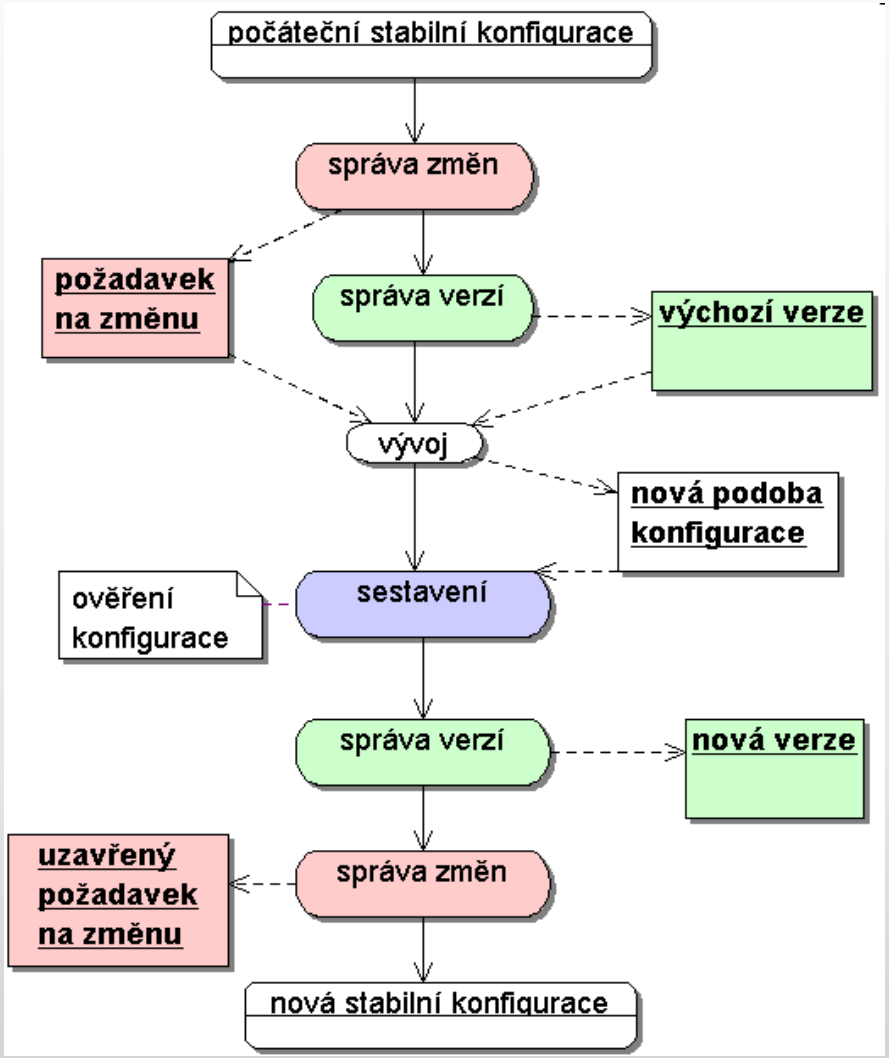


Úlohy SCM

- Určení a správa konfigurace *„cfg identification and control“*
  - určení (identifikace) prvků systému, přiřazení zodpovědnosti za správu
  - identifikace jednotlivých verzí prvků
  - kontrolované uvolňování (release) produktu
  - řízení změn produktu během jeho vývoje
- Zjišťování stavu systému *„status accounting“*
  - udržení informovanosti o změnách a stavu prvků
  - zaznamenávání stavu prvků konfigurace a požadavků na změny
  - poskytování informací o těchto stavech
  - statistiky a analýzy (např. dopad změny, vývoj oprav chyb)
- Správa sestavení (build) a koordinace prací *„release management“*
  - určování postupů a nástrojů pro tvorbu spustitelné verze produktu
  - ověřování úplnosti, konzistence a správnosti produktu
  - koordinace spolupráce vývojářů při zpracování, zveřejňování a sestavení změn

Aktivity SCM v cyklu vývoje

- Správa změn
- Identifikace a správa verzí
- Sestavení



SCM: správa změn

Change is inevitable
except from vending machines.

Správa změn

- Problém: Jak zvládat množství požadavků na úpravy produktu (opravy, vylepšení)? Jak poznat kdy už jsou vyřešeny? Jak dohledat, co bylo změněno?
  - změnové řízení, change management

- Nutný striktní postup akcí
  - vyřešení prioritních, udržení konzistence a stability
  - informovanost o změnách, prevence duplikování práce
- Význam ve všech fázích ŽC
  - evidence požadavků během jejich sběru
  - přiřazení příslušné práce během vývoje
  - hlášení a opravy chyb nalezených při testování
  - úpravy produktu během provozu a údržby

Ticket, požadavek na změnu

- Hlášení problému (bug report), feature request
  = popis nalezeného nedostatku nebo požadovaného vylepšení
  - někdy obecně zváno „ticket“
  - strukturovaný dokument, obvykle v ALM nástroji
  - zdroj: QA aktivity, uživatel, marketing

- Požadavek na změnu (change request, CR) – popisuje změny, které se mají provést na prvku/prvcích konfigurace
  - 1 ticket ↔ 1..N požadavků na změny

- Někdy (často) spojovány do jednoho dokumentu/formy

Příklad

- Hlášení problému:

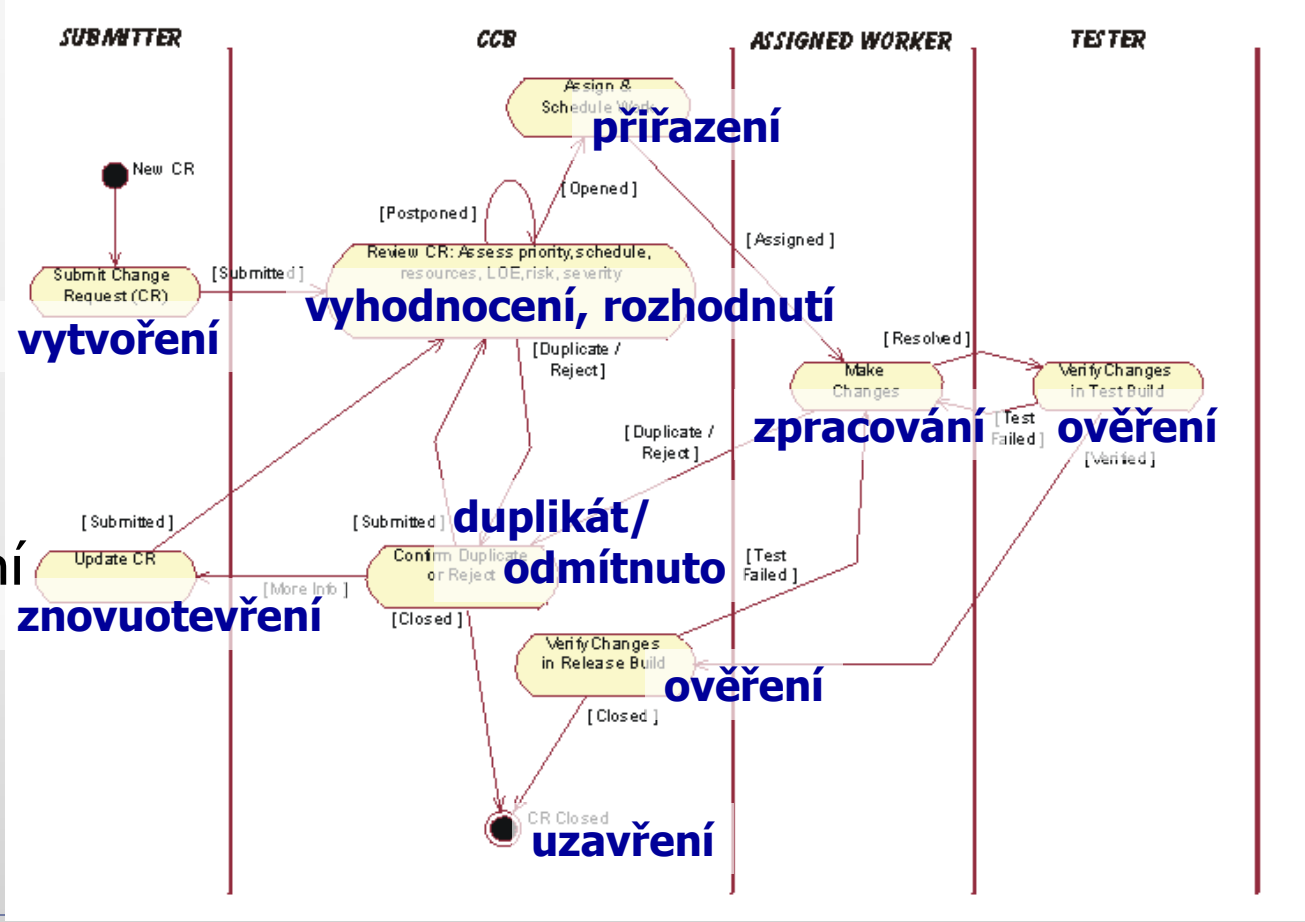
```
Package: hello
Version: 1.3-2
Severity: serious

When I invoke `hello' without arguments from an ordinary
shell prompt it prints `goodbye‘ instead of `hello, world'.

 $ hello
 goodbye

I am using Debian 1.1, kernel version 1.3.99.15z
and libc 5.2.18.3.2.1.3-beta.
```

- Požadavky na změny, vygenerované na základě hlášení:
  - *Fix standard message printout*
  - *Add tests to verify standard message is printed out correctly*
  - *Update man page*

Cyklus správy změn

- Požadavek na změnu – stavy
  - vytvořený
  - schválený
  - přiřazený
  - vyřešený
  - ověřený
  - uzavřený
  - znovuotevřený
  - odmítnutý

- Během provádění
  - znovuotevření problémů
  - vygenerování nových hlášení



Hlášení problému – detaily (1)

- Při vytvoření
  - nutné náležitosti: shrnutí (+ id, autor, datum)
  - **popis**, co nejpřesnější
    - jak chyba vznikla
    - jak reprodukovat
  - screenshot, **vzorek dat**, …
  - **závažnost**, priorita (odhad)
  - **konfigurace** daného sw a systému (OS, knihovny, …)
    - včetně identifikace verzí
- *dress code: informal × formal*

Příklad – dobrý a špatný bugreport

Hlášení problému – detaily (2)

- Při vyhodnocení
  - závažnost, **priorita**
  - **odhad** pracnosti
    - přímé (kód), návazné (doc)
  - komponenta, verze
  - **závislosti** („meta-bug“)
  - zodpovědný vývojář

Příklad – issue with discussion

- Po uzavření
  - shrnutí, zdůvodnění
  - skutečná **pracnost**
  - výsledná **revize** souborů/aplikace

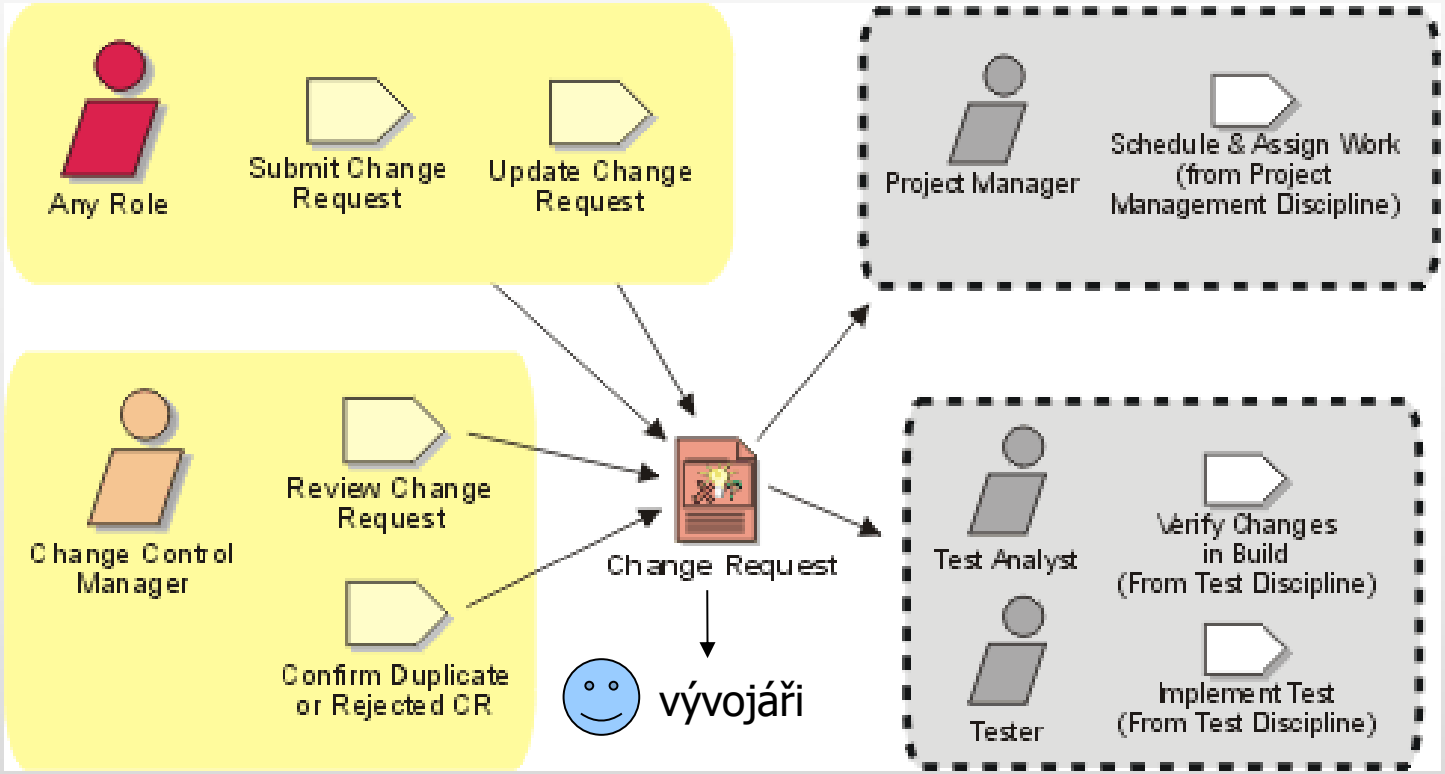
Akce při zpracování požadavku

- Vytvoření/přijetí
  - přidělení ID
- Vyhodnocení
  - možná řešení, jejich dopady a odhad pracnosti
  - doplnění popisů, meta-dat (komponenta, verze původu, …)
- Rozhodnutí
  - způsob vyřízení (vyřešit – odmítnout – duplikát – odložit)
  - závažnost (kritická chyba – problém – vada na kráse – vylepšení), priorita (vyřídit okamžitě – urgentní – vysoká – střední – nízká)
  - naplánování (iterace či cílová verze), přidělení (tým, vývojář)
- Zpracování
  - vygeneruje příslušné podřízené požadavky na změny
  - komentáře, diskuse, související sady změn (commity)
- Uzavření (nejprve všech podřízených požadavků)
  - → build: ověření konzistence + verzování: merge, někdy nový tag
  - informování zadavatele hlášení, další zájemci

Případy nouze

- Kdy porušit pravidla (proces)
  - marginální oprava těsně před release
  - vyřešení problému u zákazníka
- Pravidla pro porušování pravidel
  - je jasný (dlouhodobý) přínos výjimky
  - nekamuflovat
    - každý má vidět, že nebyl dodržen standardní proces
    - zdůvodnitelné, proč se tak stalo
  - zpětně zdokumentovat provedené akce
    - vytvořit hlášení problému, popsat provedené změny
  - opakované porušení musí vést k úpravě procesu
    - učit se z toho, co život přináší

Role ve správě změn

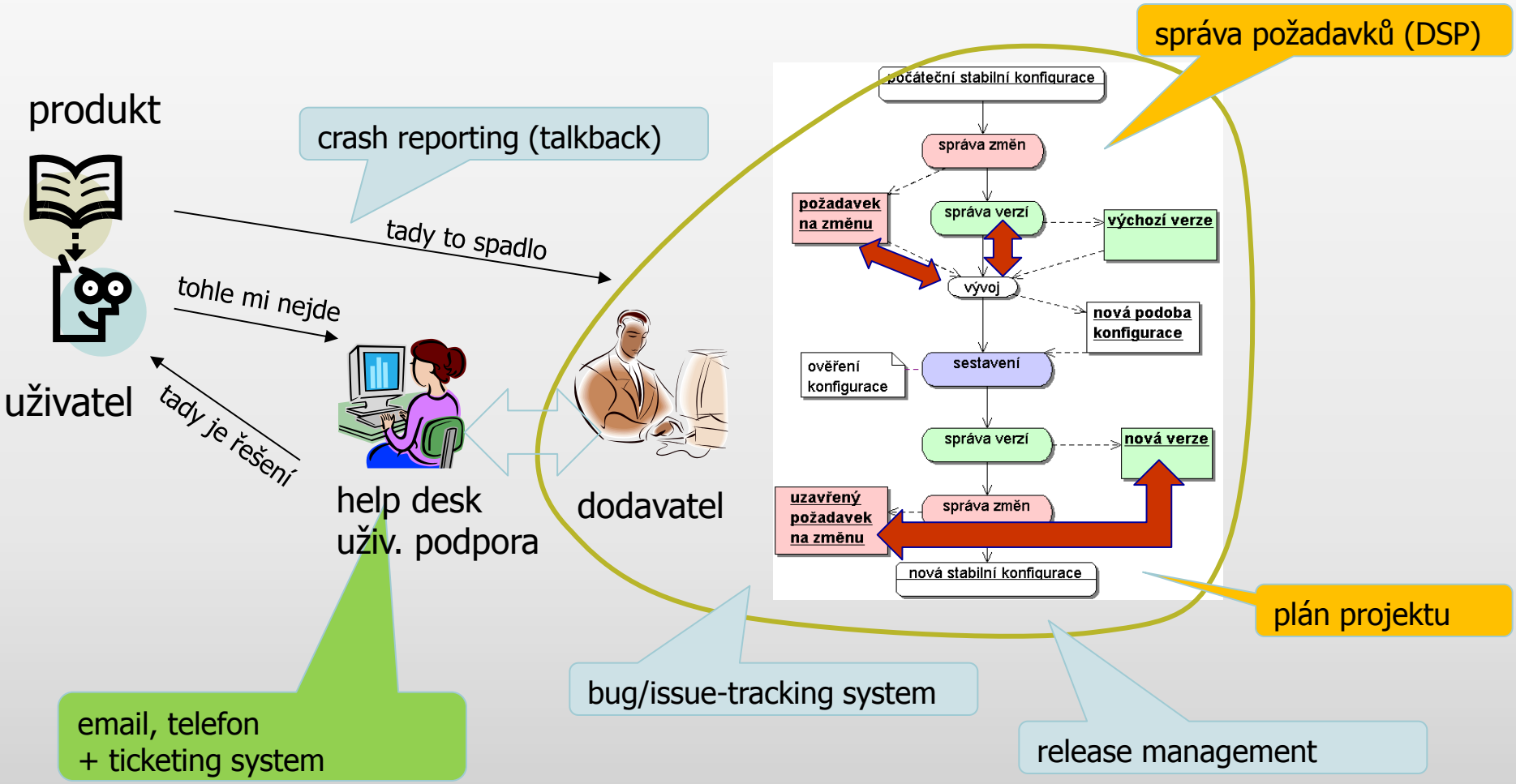


Change Control Board

- Skupina členů projektu, která má zodpovědnost za změnové řízení
  - vyhodnocování a schvalování hlášení problémů
  - rozhodování o požadavcích na změny
    - CCB může významně ovlivňovat podobu a chod projektu
  - sledování hlášení a požadavků při jejich zpracování
  - koordinace s vedením projektu
- Složení CCB
  - jedinec – vývojář, QA osoba
  - tým – technické i manažerské role
    - vhodné pokud má změna mít velký dopad
    - znalost účelu produktu

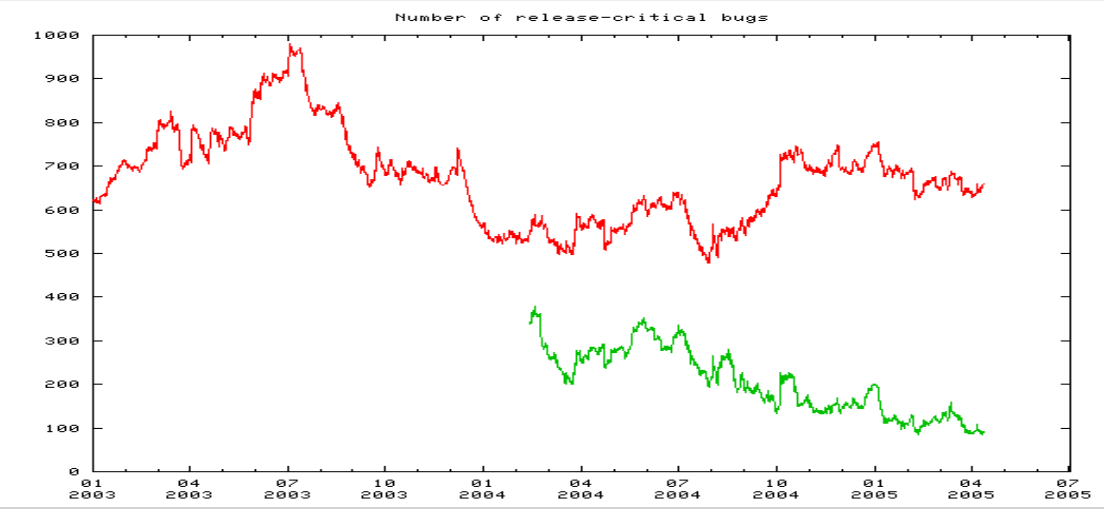
Souvislosti správy změn

Správa změn není izolovaná aktivita



Správa změn a řízení projektu

- Kritéria dodávky (release management)
  - časový termín
  - implementovaná funkčnost / vlastnosti
  - kvalita produktu
- Jak určit termín podle kvality?
- Konverguje nám iterace?



By Severity	Open	Resolved	Closed	Total
feature	463	62	581	1106
trivial	30	6	92	128
text	24	7	92	123
tweak	58	12	152	222
minor	318	132	721	1171
major	116	54	414	584
crash	12	7	50	69
block	7	12	89	108

Pamatujete backlog, burndown?

Správa změn a verzování

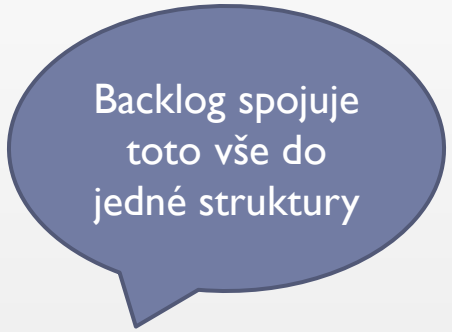
- Cíl: trasovatelnost
- Vyhodnocení požadavku: jaké verze se týká
  - uživatelská verze
  - interní verze ze správy verzí
- Uzavření požadavku
  - do BT výsledou verzi
  - do správy verzí ID vyřešeného požadavku

- Vazba ticket-commit klíčová
  - viz vzor „Task-level commit“

```
Fixed in CVS.

core/print_api.php ->1.126
core/project_hierarchy_api.php ->1.5
core/user_api.php -> 1.95
graphs/graph_by_release_delta.php -> 1.9
```

Správa změn a požadavky

- Požadavek = feature request
- Workflow
  - vize
  - požadavky → DSP
  - feature requests → bugtracker
  - (testy)
  - bug report a/nebo re-open feature request
- Flexibilita, souvislost s plánováním



Samostudium: UP Guideline: Managing Work Items

Správa změn a údržba

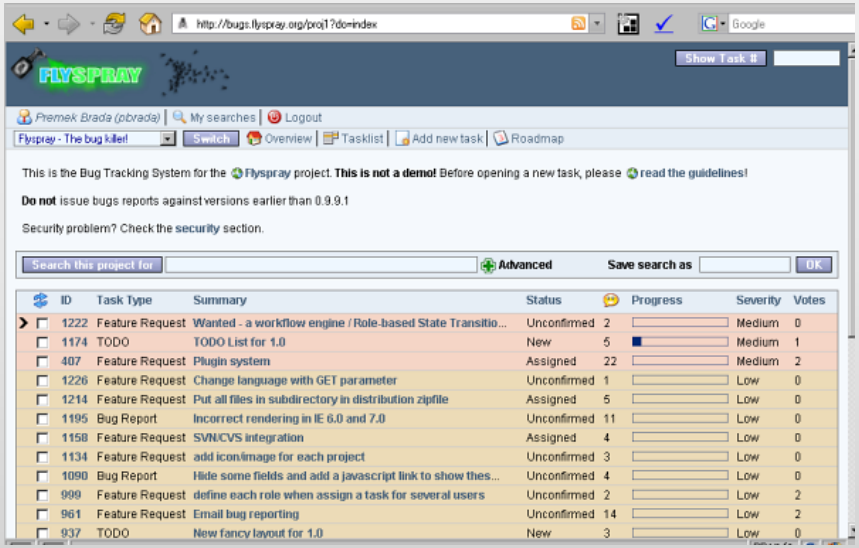
- Provoz produktu (uživatelská podpora)
  - vs. aktivní vývoj
  - nutná o to větší pečlivost při úpravách
- Hlášení problému
  - oprava
  - garance, servisní smlouva / vícepráce
- Požadavek na vylepšení
  - ihned => vícepráce
  - naplánovat do přírůstku

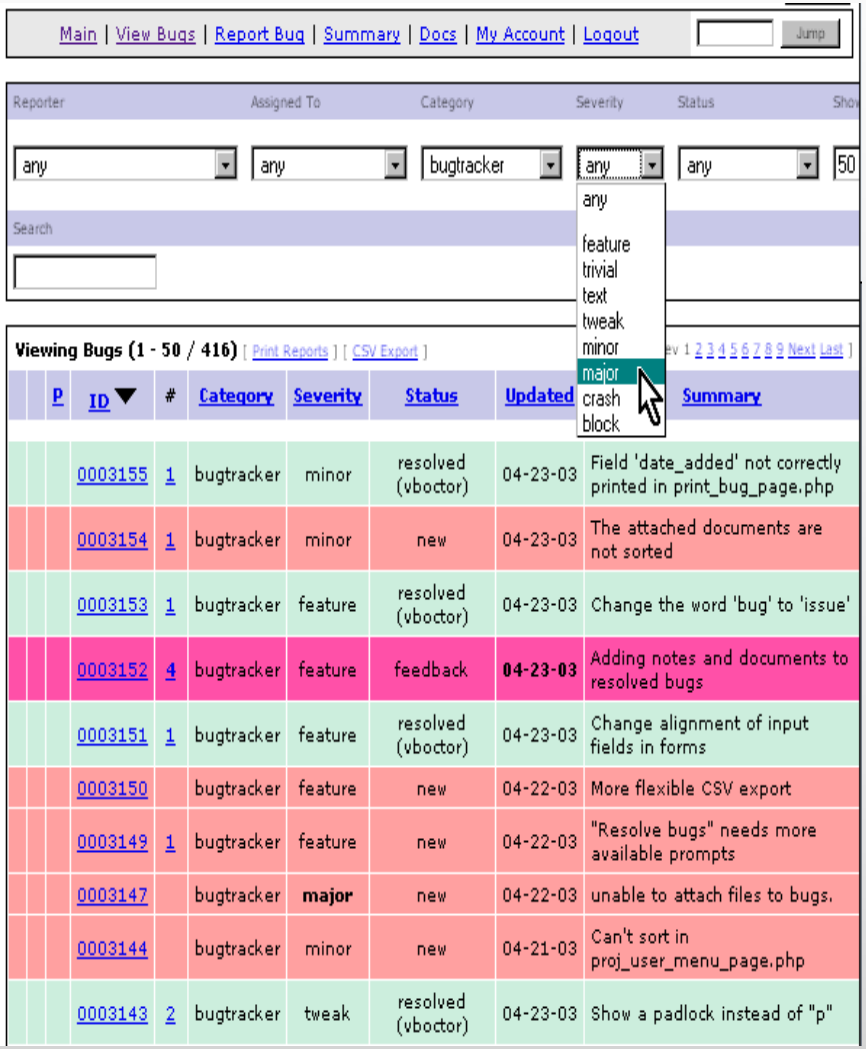
Systémy pro správu změn

- Bug tracking (BT) systémy
  - evidence, archivace požadavků
  - skupiny a uživatelé
  - kategorie hlášení
  - úrovně závažnosti, priority
  - verze produktu, operační systém
  - vyhledávání
  - reporty, grafy, statistiky
  - sledování stavu požadavku
  - cfg workflow, reportu
- ALM (Application Lifecycle Management)
  - Bug/issue tracking funkčnosti a charakteristiky (viz vlevo)
  - Odhad a realita pracnosti
  - Plánování – milníky, fáze/iterace
  - Sledování – metriky, grafy
  - Těsná vazba na VCS
  - Dokumentace a komunikace (wiki, dokumenty, notifikace)
  - Multi-team a multi-project
  - Integrace s IDE

BT systémy: Flyspray, Mantis, Bugzilla

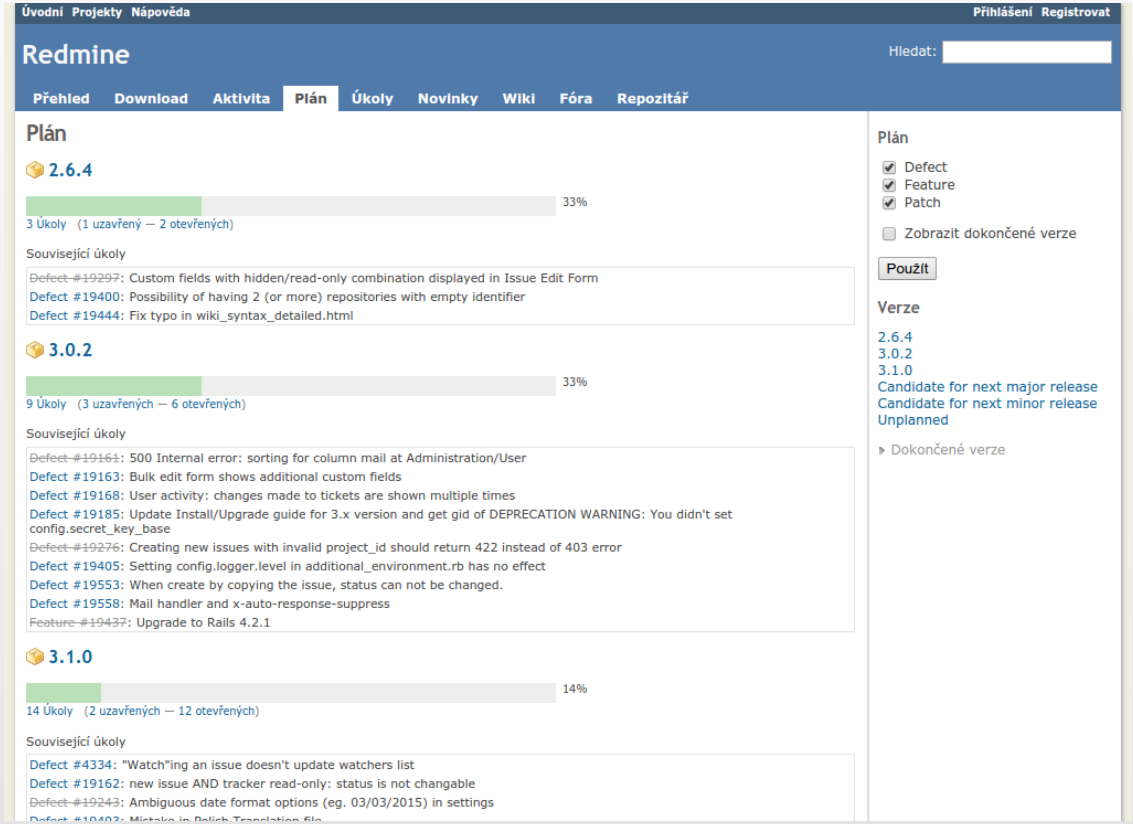
- Spíše jednoduché
- Snadná instalace
- Webové rozhraní
  - emailová notifikace





ALM: Redmine, Jira+Confluence, RTC

- Robustní
- Velké projekty
  - apache.org
  - firemní prostředí
- Konfigurovatelné



Příklad – vyhledávání > seznam > detail > historie

SCM: správa verzí

Správa verzí

- Součást úlohy SCM „identifikace konfigurace“
  - aby prvek konfigurace mohl být ve správě SCM, musí být identifikovatelný, včetně všech svých podob
- Účel správy verzí: udržení přehledu o podobách prvků konfigurace a konfigurací

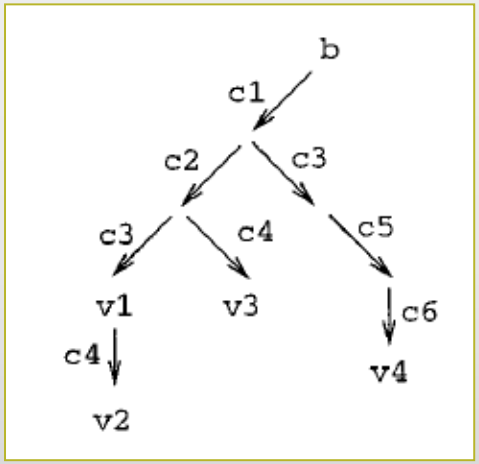
- Verze = jedna konkrétní podobu prvku nebo konfigurace
  - jaké aspekty zahrnuje „podoba“ => druh verzování
  - co je verzováno => granularita verzování
  - jak je verze určena (identifikace) => typ verzování
- Nástroje: úložiště a verzovací systém (VCS)

Doporučená četba: R Conradi: Version models for software configuration management (1998)

Určení konkrétní verze prvku

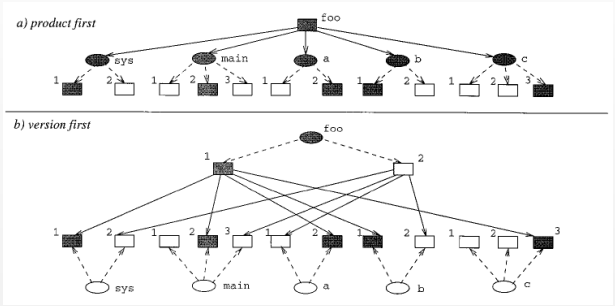
- Základní druhy verzí
  - historická podoba → revize („Word 6.0“)
  - alternativní podoba → varianta („Word pro Macintosh“)

- Verzování podle stavu
  - identifikují se pouze podoby prvků
  - výsledná verze vznikne vhodným výběrem
- Verzování podle změn
  - identifikují se (také) změny prvků = changeset
  - výsledná verze vznikne aplikací změn

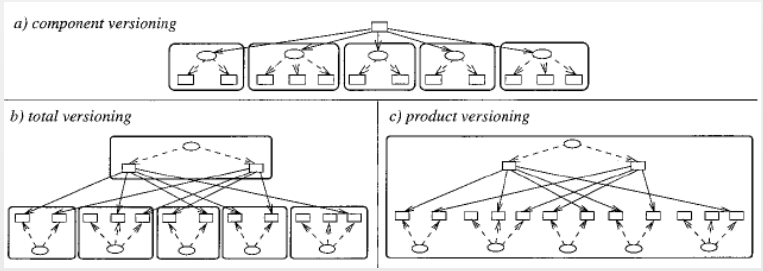


Granularita verzování

- Product Space x Version Space



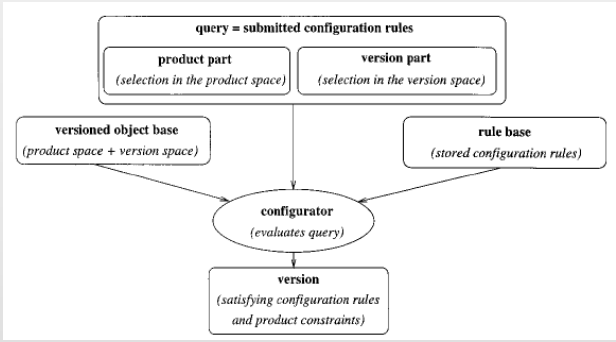
- verzování komponent
  - jednotlivé prvky samostatně
  - konfigurace nemá verzi
- úplné verzování
  - verzi má celá (sub)konfigurace
  - indukuje verze prvků



- produktové (uniformní) verzování
  - struktura (sub)konfigurace a systém verzování nezávislé
  - výběr verze indukuje prvky konfigurace

Identifikace konkrétní verze

- Extenzionální verzování
  - každá verze má jednoznačné ID
    - např. 1.5.1 = verze pro DOS, 1.5.2 pro UNIX,
      2.1.1 = nový release pro DOS
  - jednoduchá implementace
    → problémy při větším počtu verzí
  - naprostá většina VCS



- Intenzionální verzování
  - verze je popsána souborem atributů
    - např. OS=DOS and UmiPostscript=YES
  - nutné pro větší prostory verzí → potřeba vhodných nástrojů

Informace o verzi

- Identifikátor verze (extenzionální)
  - klíčový požadavek: jedinečnost
  - schémata
  - „major.minor.micro + build“ – např. 6.0.2800.1106 (MSIE 6)
    - význam pozic identifikátoru a změny hodnot
    - standardní sémantika: kompatibilita
  - dle nástroje – např. $Revision 1.1.2.1$ v CVS
    - pro prvky konfigurace
  - „marketingové“ jméno – např. “One tree hill ” (Firefox 0.9)
    - pouze pro produkt

http://semver.org/

- Meta-data
  - datum/čas vytvoření, autor
  - stav prvku/konfigurace
  - předchůdce (předchůdci)

Nástroje pro verzování

- Úložiště (databáze projektu, repository) = sdílený datový prostor, kde jsou uloženy všechny prvky konfigurace projektu
  - centrální místo
  - určitě všechny prvky ve všech verzích
    (způsob uložení: dle granularity a typu verzování)
- Nutný řízený přístup (udržení konzistence)

⇒ Version Control System (VCS)

- sada nástrojů pro práci s úložištěm

Implementace:
• souborový systém + dohodnutá pravidla používání
• verzovací nástroj
• databáze s rozhraním podporujícím postupy SCM

Práce s úložištěm

- Základní operace
  - *vytvoření* – založení úložiště, případně zpřístupnění existujícího
  - *inicializace* – naplnění boostrap verzí projektu
  - *check out* – kopie prvku do lokálního pracovního prostoru
  - *check in* (commit) – uložení změněných prvků do úložiště
  - *branch* + *merge* – oddělení různých prací
  - *tag* – označení vybrané verze symbolickým identifikátorem
  - *zjišťování stavu* – sledování změn v úložiště vs. prac.prostor

- Pozn: přístup k zamykání při ci/co
  - read-only pro všechny
  - pesimistický: read-write kopie prvku jen pro pověřeného
  - optimistický: read-write pro kohokoli, řešení konfliktů

Centrální vs distribuovaný vývoj (1)

- Potřeby: Je tým geograficky distribuovaný? V čase distribuovaný? Potřebují členové pracovat offline (a synchronizovat)? Musí existovat „jedna sdílená pravda“? Chci snadný „fork“ projektu?

- Možnosti
  - centrální úložiště
  - privátní větve
  - distribuovaný verzovací systém
- Typické přístupy
  - CVS, Subversion x git, RTC
  - unikátní ID commitu, HEAD vs TIP

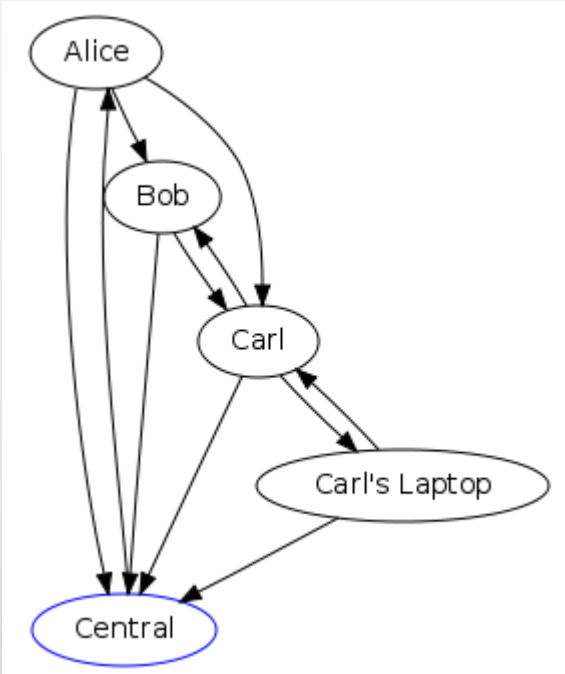


Image source: http://mercurial.selenic.com/wiki/UnderstandingMercurial

Centrální vs distribuovaný vývoj (2)

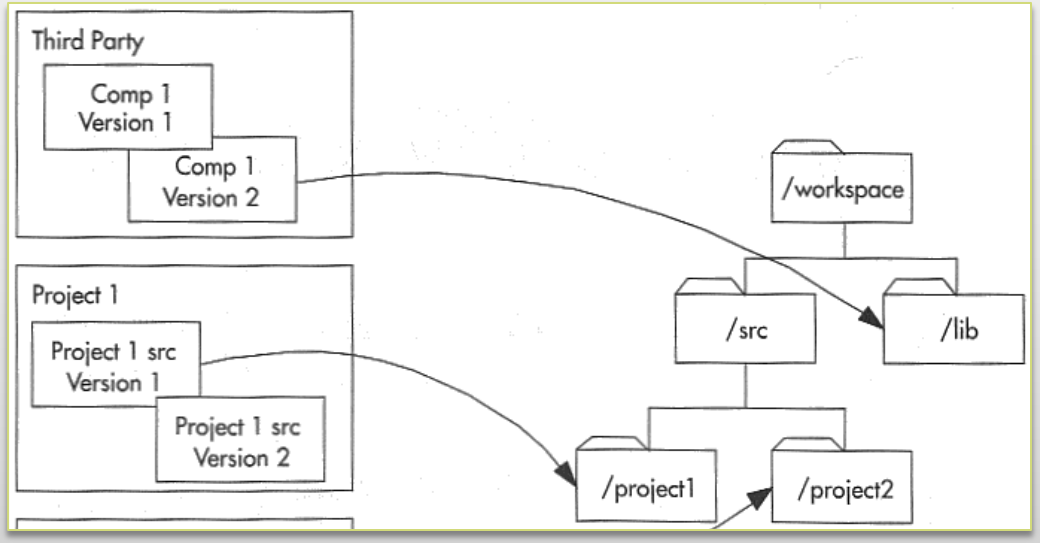
- Vede na různé způsoby práce
- Centralizovaný
  - update
  - commit
  - Vyžaduje online přístup, zodpovědnost; poskytuje jednoduchost a přehled
- Distribuovaný
  - fetch, update, commit, cleanup
  - push
  - Vyžaduje znalosti a disciplínu, rozlišení zdrojového úložiště; poskytuje variabilitu procesu

nezaměňovat „distribuovaný“ a „vymoženosti git“

Doplňující info: http://mercurial.selenic.com/wiki/UnderstandingMercurial

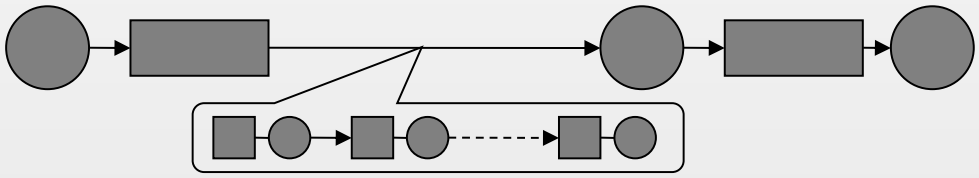
Pracovní prostor

- Pracovní prostor (workspace) = soukromý datový prostor, v němž je možno provádět změny prvků konfigurace, aniž by byla ovlivněna jejich „oficiální“ podoba používaná ostatními vývojáři
  - vývojářský (soukromý)
  - integrační (sdílený)
- Zpřístupnění úložiště pro práci
  - *checkout*
  - *clone*

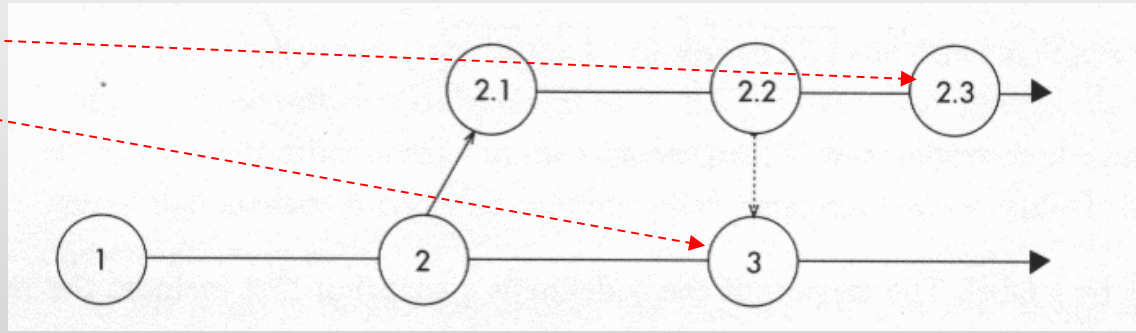


Terminologie: Codeline

- Codeline (vývojová linie) = série podob (verzí) množiny prvků konfigurace tak, jak se mění v čase
  - (extenzionální verzování podle stavu) obsahuje commity a/nebo changesety, větve => část grafu verzí

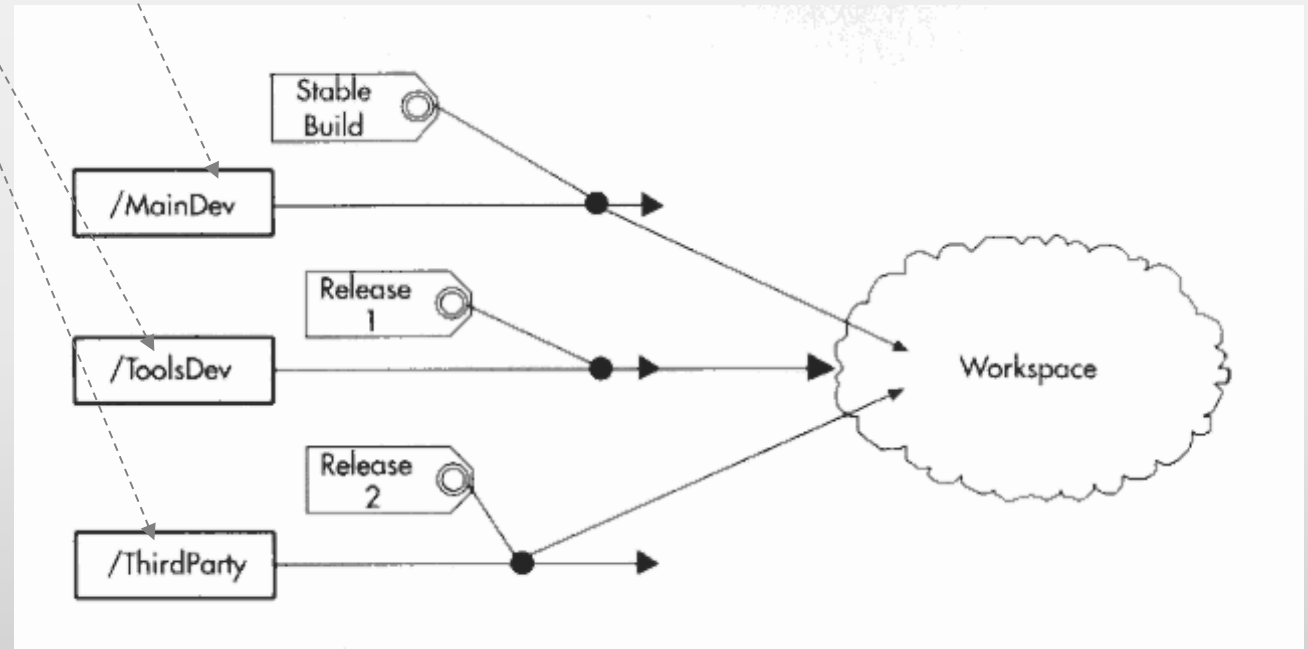


  - vrchol codeline obsahuje nejčerstvější verzi
    - "HEAD", "tip"
  - má přiřazena pravidla práce na codeline



Codeline (2)

- Konfigurace se může skládat z více codelines
  - vlastní projekt
  - knihovní codeline



Pravidla na codeline

- Codeline má přiřazena pravidla práce (policy)
  - klíčová součást (viz architektura – konvence)
  - odlišení různých codelines

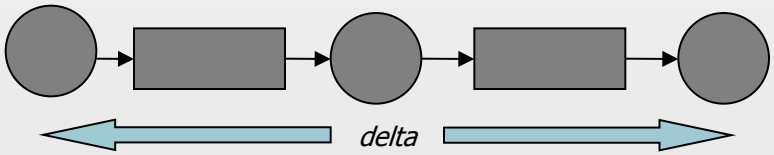
- typ prací (vývoj, údržba, release, …)
- očekávaná kvalita kódu
- pravidla/akce před ci, po checkout, …
- jak a kdy ci, co, branch, merge
- přístup pro osoby a skupiny
- kam (codeline) exportovat změny, odkud importovat
- doba platnosti či podmínky odstavení

Příklad – release branch (viz dále)

Delta, diff

- Delta = množina změn prvku konfigurace mezi dvěma po sobě následujícími verzemi
  - příklad: přidání sekce „kontext produktu“ do DSP, patch foo.c
  - v některých systémech jednoznačně identifikovatelná
  - changeset: delta + důvod

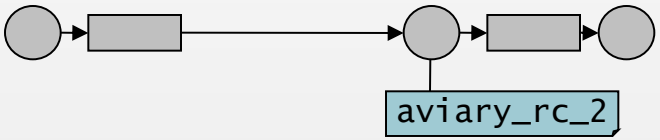
diff* a *patch
- rozdíl mezi verzemi (text, binární)
- aplikace rozdílu na verzi
  - viz changeset



Příklad – RCS *,v + diff

Tag, baseline

- Tag, label = označení konfigurace symbolickým jménem

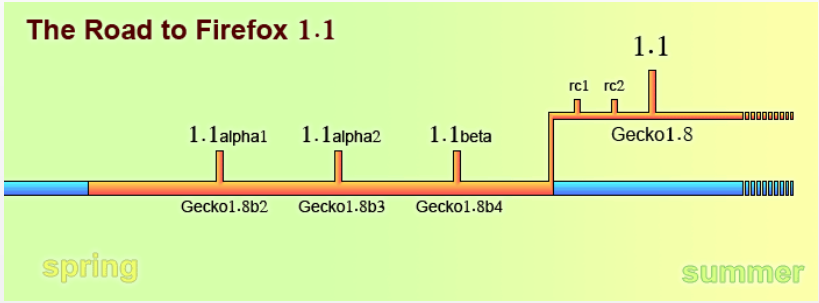


- Baseline = konzistentní konfigurace tvořící stabilní základ pro produkční verzi nebo další vývoj
  - příklad: milník „stabilní architektura“, beta verze aplikace
  - stabilní: vytvořená, otestovaná, a schválená managementem
  - změny prvků baseline jen podle schváleného postupu

- Při problémech návrat k tag, baseline

Význačné baseline

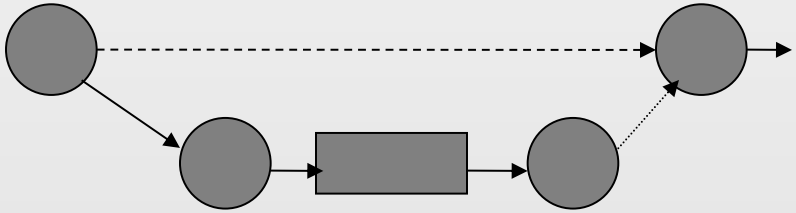
- Interní release
  - iterace
- Milníky projektu
  - jednotlivé fáze životního cyklu
    - vodopád / spirála / iterativní
  - alfa verze = „feature complete“, interní testování
  - beta verze = testování u (vybraných) zákazníků
- Finální release produktu
  - verze dodané zákazníkovi/na trh



http://www.mozilla.org/projects/firefox/roadmap-1.5.html

Paralelní práce na „stejné“ konfiguraci

- Důvod: velké úpravy, release, spekulativní vývoj, varianty, ...
  - viz SCM patterns později
- Cíl: vzájemná izolace paralelních prací tak, aby ukládané změny během nich neovlivnily ostatní
  - oddělení paralelních vývojových linií

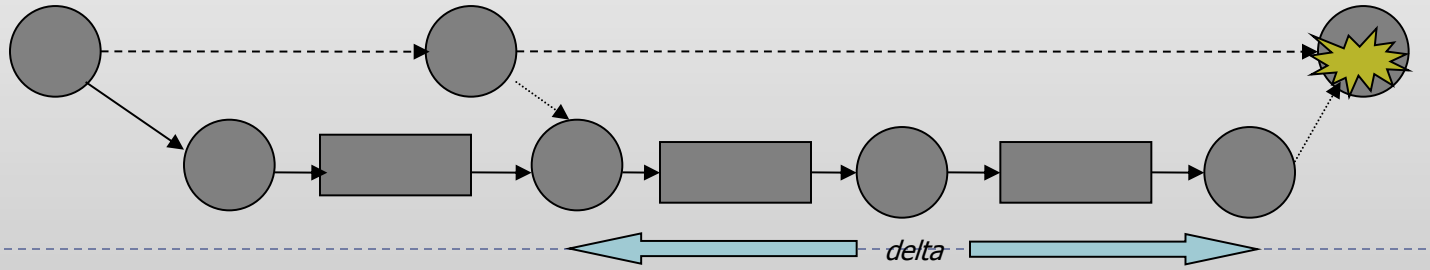


- Cena za izolaci od změn = řešení konfliktů

Větvení a spojování

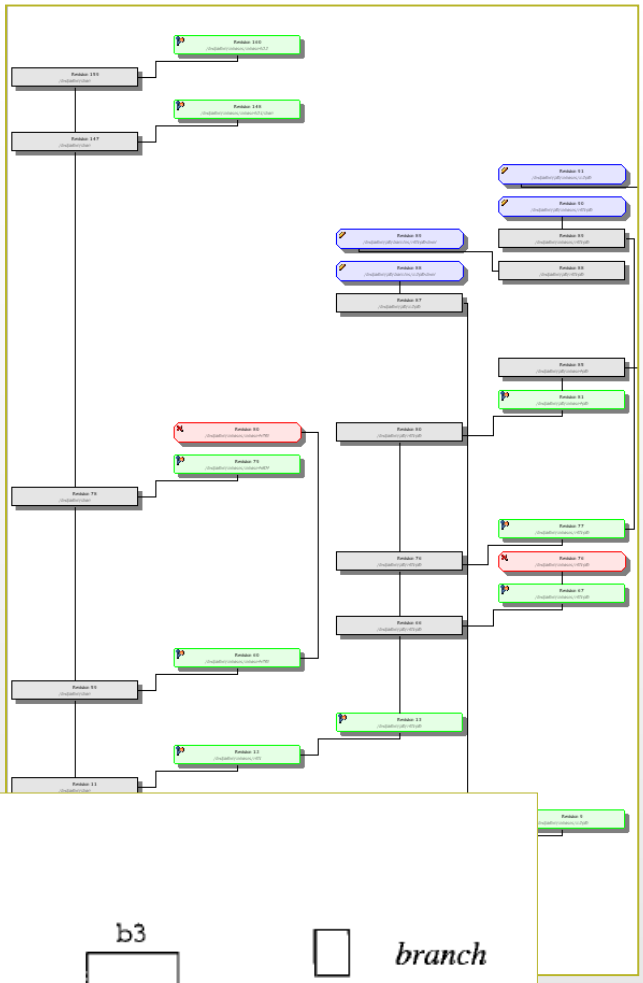
- Kmen (trunk, master) – hlavní vývojová linie
- Větev (branch) – paralelní vývojová linie
  - konkrétní účel viz vzory pro verzování
  - operace vytvoření větve (branch-off, split)

- Spojení (merge) – sloučení změn na větvi do kmene
  - slučuje se delta od branch-off nebo posledního merge
  - řešení konfliktů: automatizace vhodná, ale ne vždy možná
  - 2-way a 3-way merge

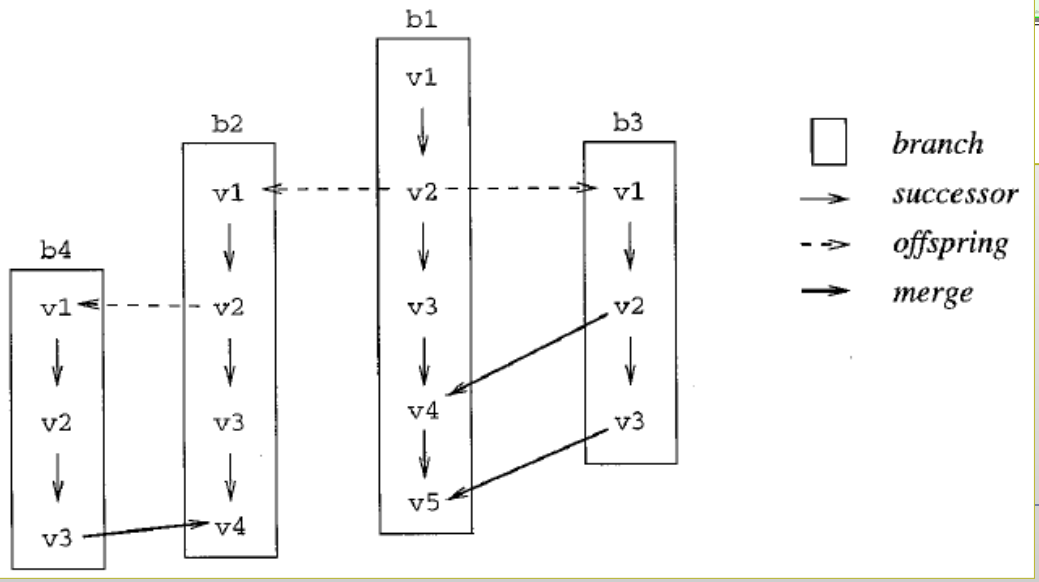


Graf verzí

- Historie vývoje konfigurace
- Zobrazení vazeb mezi verzemi
  - uzly = verze, hrany = vazby verzí
  - grafická podoba codeline

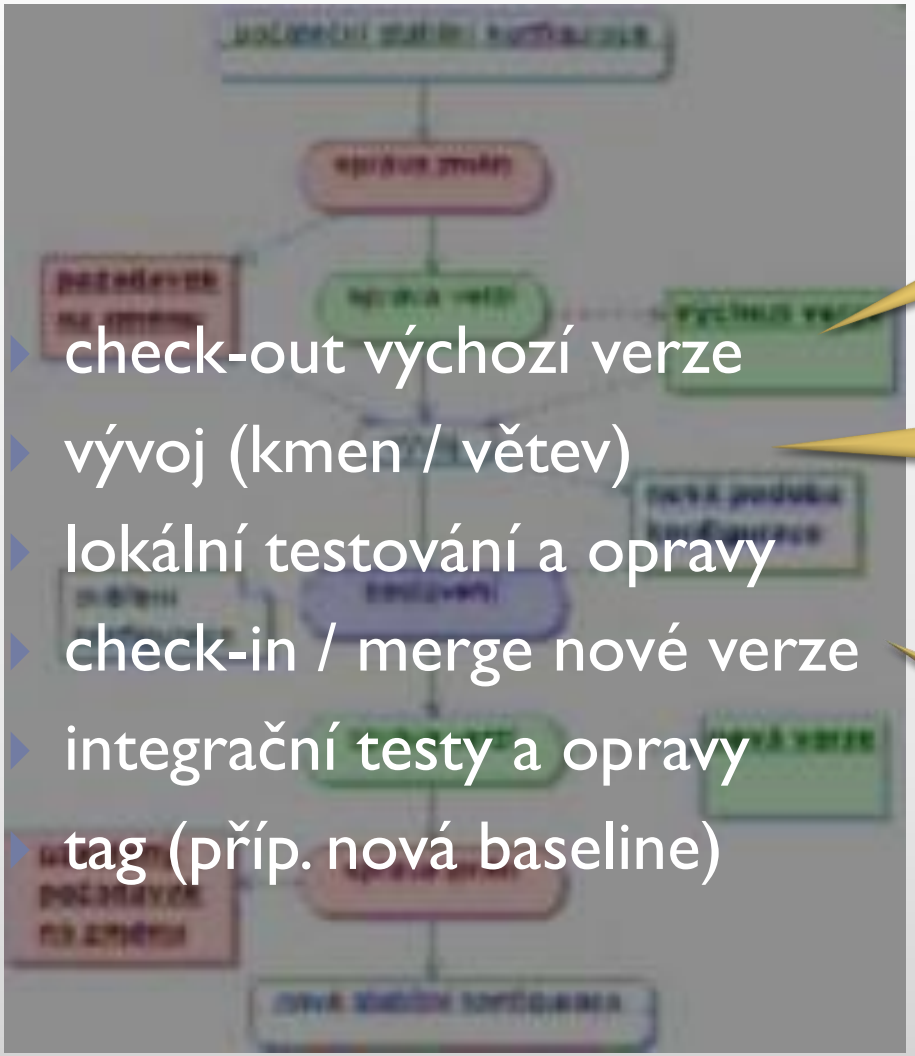


```
* c47b86b - (origin/master) Boulot du collègue sur master (Chris
| *   2e6104b - (HEAD, master) Merge branch 'feature' (Christoph
| |\
|/ /
| * a1b0ecb - (origin/feature, featu
| * d384a57 - Migrating HTML, CSS an
| * dc47a96 - README.md (Christophe
| * dca0784 - Boulot du collègue (Ch
| * 80b0beb - Footer tweak + MIT lic
|/
* 6d731f3 - Content tweaks (Christop
```



Příklad – svn / git graf

Postupy při verzování



- check-out výchozí verze
- vývoj (kmen / větev)
- lokální testování a opravy
- check-in / merge nové verze
- integrační testy a opravy
- tag (příp. nová baseline)

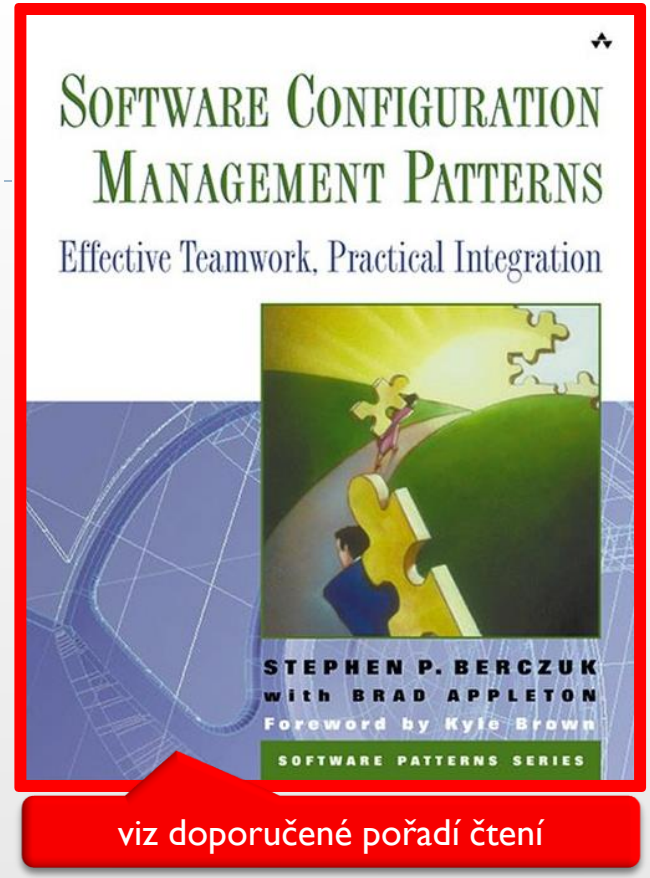
Které? Viz správa verzí:
- výchozí pro vývoj
- starší release pro bugfix

Kolik? Viz změnové řízení:
- commit per task
- commit per bug/request
- lokální commity

Kam? Viz správa verzí:
- hlavní vývojová linie
- větev (release, task)

Vzory pro verzování (1)

- Hlavní vývojová linie (*mainline*) a pravidla linie (*codeline policy*)
  - jedna codeline pro průběžný vývoj
- Stabilizační období (*code freeze*)
  - pravidla před release



viz doporučené pořadí čtení

- Větev pro release a jeho přípravu (*release line*)
  - místo code freeze mít samostatnou větev pro release
- Kód třetích stran na větev (*third party codeline*)
  - vlastní větev pro každý kód od externího dodavatele

Vzory pro verzování (2)

- Privátní verze (*private versions*)
  - soukromé úložiště pro častější check-in
- Větev pro úkol (*task branch*)
  - složitější úpravy s většími následky dělat na větvi
  - vazba na správu změn
- Check-in pro každý úkol (*task-level commit*)
  - minimum nutného
  - po ukončení práce na jednom úkolu udělat check-in změn

A successful Git branching model
http://nvie.com/posts/a-successful-git-branching-model/

Nástroje pro verzování (VCS)

Version Control Systems

- 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 ci/co prvků z repository do nástrojů
  - příklad nástrojů: ClearCase, Adele

Co nástroj má umět

- Operace s úložištěm
  - ci, co, add, rename, move, import, export
  - stav prvků, diff, historie změn
- Verzování
  - ci, co, data revize (klíčová slova)
  - branch, merge, značkování
- 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: Revision Control System

- Správa verzí pro jednotlivé textové soubory
  - UNIX, Windows, DOS
  - extenzionální stavové verzování komponent
- Ukládá (do foo.c,v souboru)
  - => historii všech změn v textu souboru
  - informace o autorovi, datumu a času změn
  - textový popis změny zadaný uživatelem
  - další informace (stav prvku, symbolická jména)
- Používá diff(1) pro úsporu místa
  - poslední revize uložena celá
  - => předchozí pomocí delta vygenerované diff-em
- Funkce
  - zamykání souborů, poskytování R/O kopií
  - symbolická jména revizí, návrat k předchozím verzím
  - možnost větvení a spojování změn z větví do kmene
- Složky (utility z příkazové řádky):
  - ci, co, rcs, rlog, rcsdiff, rcsmerge

```
head            1.3;
access;
symbols
VERZE_1_1:1.2;
locks; strict;

1.3
...

1.2
date            2004.01.09.16.44.38;
author brada;                   state Exp;
branches;
next            1.1;
...

1.2
log
@- upraveny sablony na XHTML+CSS2 layout
- pridana fce KivPage.setLang()
- drobne upravy Database.class (cfg hodnoty, retu
@
text
@d10 1
a10 1
 * @@version $Id: sitecfg.inc,v 1.1 2003/12/15 1
d41 2
```

http://www.gnu.org/software/rcs/

cpp: Realizace variant

- C preprocessor umožňuje intenzionální stavové verzování
- Např. chceme variantu foo.c pro případ OS=DOS and UmiPostscript=YES :

```
/* vlozime definice varianty */
#include "sys-variant.h"
...
#if (defined OS_DOS)
#if (defined UmiPostscript)
... /* zdrojovy kod, který odpovida variante */
#else
... /* varianta OS=DOS, UmiPostscript=NO */
#endif
... /* varianta OS != DOS */
#endif
```

- Definice atributů pro popis variant
  - hlavičkový soubor (centrální místo def. varianty celého systému)
  - parametry příkazové řádky gcc -DOS_DOS (např. v Makefile)

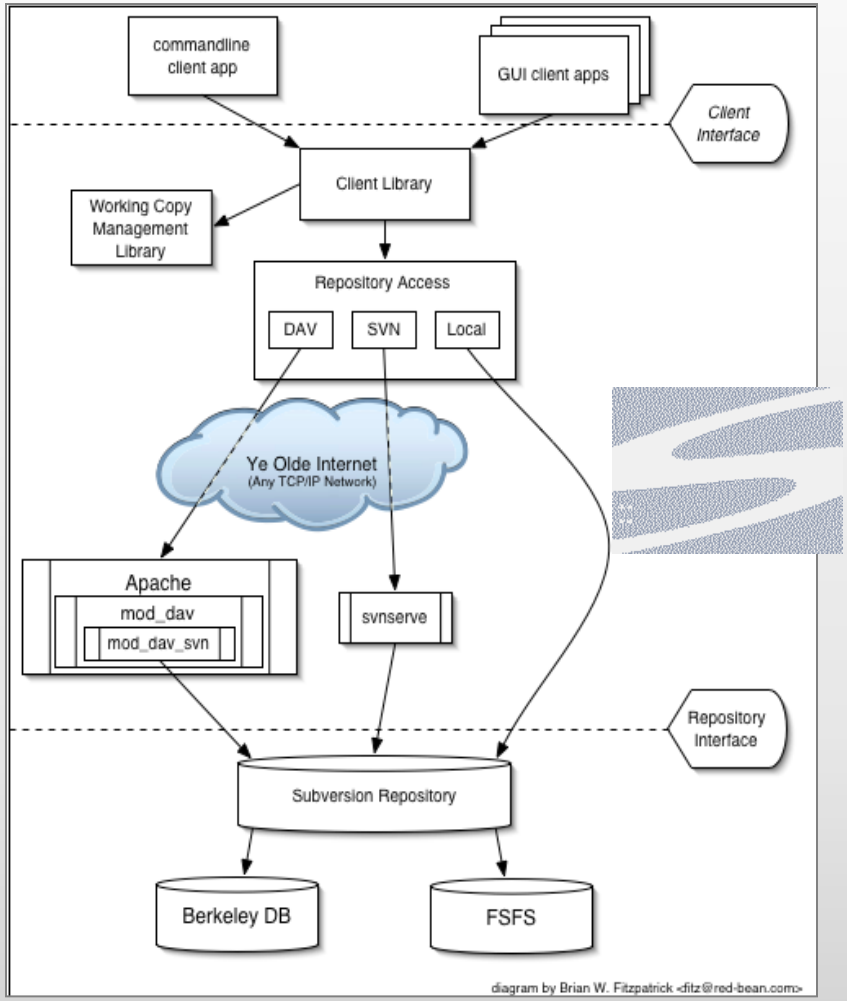
CVS (Concurrent Versioning System)

- Práce s celými konfiguracemi (projekty) najednou
- Sdílené úložiště + soukromé pracovní prostory
  - úložiště lokální nebo vzdálené (rsh/ssh, p-server)
  - optimistický přístup ke kontrole paralelního přístupu
    - místo zamkni-modifikuj-odemkni (RCS) pracuje systémem zkopíruj-modifikuj-sluč
  - zjišťování stavu prvků, rozdílů oproti repository
  - možnost definovat obsah a strukturu konfigurace
  - triggery
  - vše co umí rcs (zejm. klíčová slova)
- Free software
  - původně nadstavba nad rcs
    - používá ,v formát souborů
  - příkazová řádka, grafické nadstavby (UNIX, Windows, web)
  - integrace do mnoha IDE a CASE nástrojů

- http://savannah.nongnu.org/projects/cvs/

Subversion (svn)

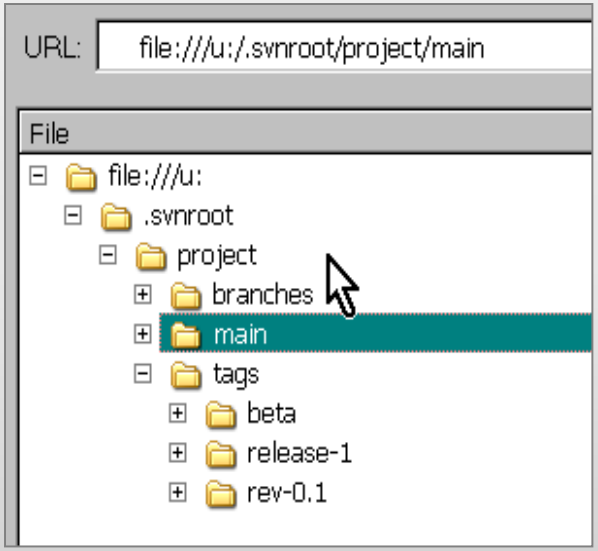
- Následník CVS
  - Karl Fogel (autor Open Source Development with CVS) najatý CollabNet v r. 2000 na vytvoření „lepšího“ CVS
  - bez omezení předchůdce – přejmenování, verzování adresářů, atomický commit, http přístup
  - nové možnosti – binární diff, meta-data, abstraktní síťová vrstva (DAV), čisté API
  - způsob práce a příkazy velmi podobné CVS



http://subversion.apache.org/

svn: Několik poznámek

- Identifikace verzí
  - globální kontinuální jednočíselné identifikátory (číslují commit)
  - není koncept „značek“ (tag) jako v CVS
- Obecná operace „copy“
  - kopíruje jednu část úložiště/projektu na jiné místo v úložišti
  - význam kopie dle potřeby => název
    - značky – označení aktuální/vybrané verze
    - vytvoření větve
- Doporučená struktura úložiště
  - oddělené adresáře pro kmen, větve a značky



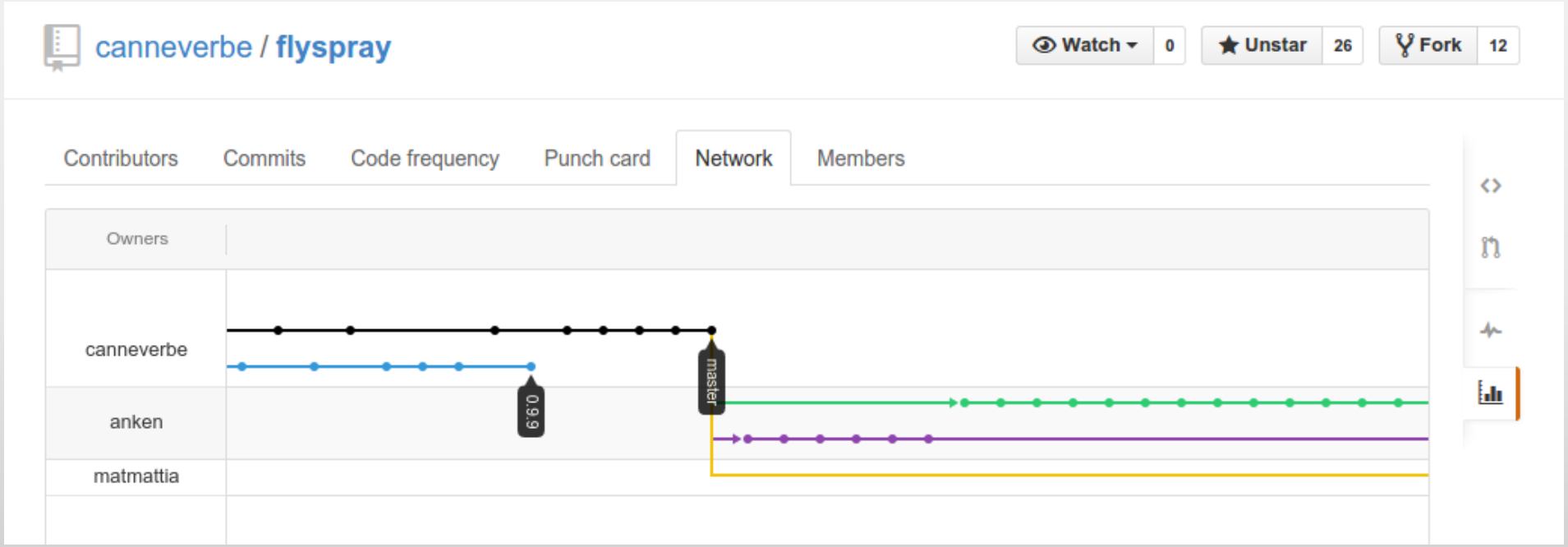
Git

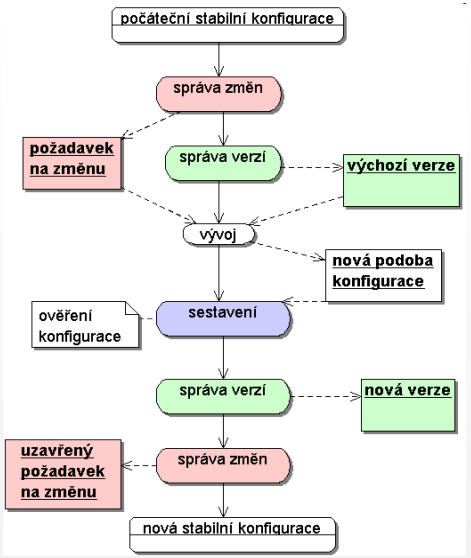
- Distribuovaný verzovací nástroj
  - původ: Linus Torvalds, správa linux kernel patches
- Repository
  - master, lokální
  - origin
- Operace
  - clone, pull, push
  - rebase, revert, cherry-pick, stash

http://git-scm.com/

GitHub

- nadstavba git
- hosting úložišť, jednoduchý project management
- „social coding“
  - velmi snadné sdílení příspěvků: pull request





SCM: řízení sestavení (build)

Řízení sestavení

- Aktivity provádějící transformaci zdrojových prvků konfigurace na odvozené
  - zejména sestavení celého produktu
- Cíl: vytvořit systematický a automatizovaný postup

- Pojmy
  - build (též integration; proces sestavení, sestavení) – proces a výsledek vytvoření částečné nebo úplné podoby aplikace
  - zdrojové a odvozené prvky konfigurace

Postup při vytváření sestavení

- Build process
  - míra formálnosti
  - míra preciznosti

- Kroky
  - příprava
  - check-out
  - preprocessing, překlad, linkování
  - nasazení
  - spuštění
  - testování
  - značkování, check-in
  - informování

Příklad sestavení

```
/work/prj/hello# rm -rf * .*

/work/prj/hello# svn co file:///d:/data/svnroot/hello.ukazkove/trunk/ .
Restored 'Makefile'
A    lib
A    src
A    src\hello.c
A    src\vypisy.c
A    src\vypisy.h
A    bin
U    Makefile
Checked out revision 46.

/work/prj/hello# make
gcc  src/*.c -o bin/hello
Preklad OK, spustis pres "make run“

/work/prj/hello# make test
*** Test spusteni
bin/hello.exe
Hello, world!
*** Test napovedy
bin/hello.exe -h
Program Hello verze 1.
Vypisuje uvitaci zpravu.
Volani: hello

/work/prj/hello# svn ci -m “TAG Working version 1“
   file:///d:/data/svnroot/hello.ukazkove/trunk/
   file:///d:/data/svnroot/hello.ukazkove/tags/v1-working-20090412/
```

Vlastnosti sestavení

- Jedinečnost a identifikovatelnost
  - PROJEKT_v2_build2134_20041220T1954
  - identifikátor jednoznačný, čitelný
  - vytvořitelný a zpracovatelný automaticky (schema pro id)
- Úplnost
  - tvoří kompletní systém, obsahuje všechny komponenty
- Konzistence
  - vzniklo ze správných verzí správných komponent
    - tj. z konzistentní konfigurace
- Opakovatelnost
  - možnost opakovat build daného sestavení kdykoli v budoucnu
    - se stejným výsledkem
- Dodržuje pravidla vývojové linie
  - build odpovídající baseline
  - zejména release má striktní pravidla

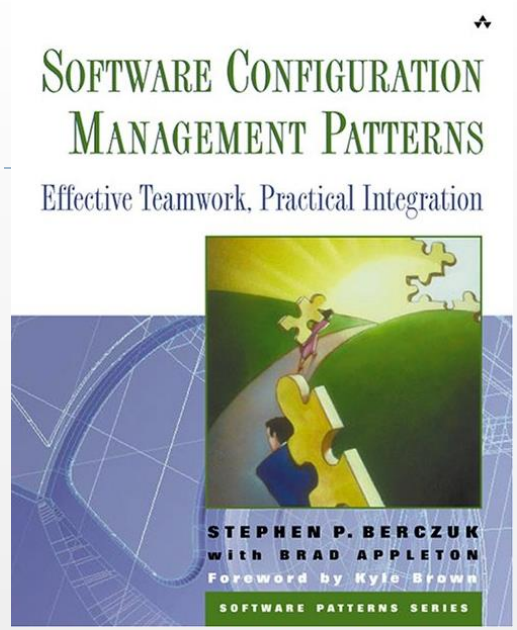
Součásti prostředí pro sestavení

- Pravidla (neměnit)
  - vývojová linie
  - součásti a vlastnosti sestavení
    - adresářová struktura, identifikátory sestavení
- Scripty
  - check-out, značkování, check-in
  - preprocessing, překlad, linkování
  - nasazení, spouštění, testování
    - některé nebudou u interpretovaných jazyků, dokumentů apod.
  - informování vývojářů, vytváření statistik
  - vytvoření distribuční podoby (packaging)
- Vyhrazený stroj a workspace
  - „build machine“
  - zejména pro integrační a release sestavení

Typy sestavení

- Co je použito pro sestavení
  - čas překladu x jistota správnosti
  - čistý
  - úplný
  - přírůstkový (inkrementální) build
- Účel sestavení
  - lokální/neoficiální komponenty povoleny
  - soukromý
  - integrační (oficiální)
  - release build

Vzory pro sestavení (1)

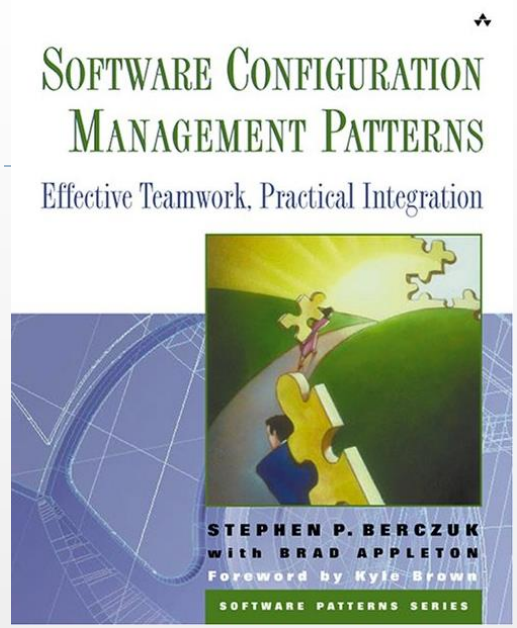


- Základní postupy
  - soukromé sestavení (*private system build*) + sdílení součástí
  - integrační sestavení (*integration build*)
  - release build
- Podpůrné aktivity
  - kusovník (Bill of Materials) a zapouzdřená identifikace
  - archivace prostředí
  - balení a distribuce (packaging)
- Obecný cíl:
  odchytit co nejdříve okamžik kdy „se to rozbilo“

Vzory pro sestavení (2)

- QA-related postupy
  - zkouška těsnosti (*smoke test*)
  - regresní testy (*regression test*)
- Typické workflow
  - *Daily Build and Smoke Test*
  - *Continuous Integration*

- Obecný cíl:
  opakovatelná záruka dostatečné kvality



Soukromé sestavení

- Cíl: ověřit si konzistenci konfigurace
  - produkt lze sestavit po mnou provedených změnách
  - před check-in (problémy řeším já x všichni)
- Postup: sestavit produkt v soukromém prostoru
  - pomocí sestavení (scriptu) co nejpodobnějšího oficiálnímu
    - postup, verze nástrojů, adresářová struktura
    - rozdíly => problémy
  - obsahuje všechny závislé součásti produktu
  - ze zdrojových textů zejména změněné a přímo související
    - ze správy vzít pro build verzí vše / kromě označeného / pouze označené
- Urychlení průběhu
  - použít inkrementální sestavení, kde je to vhodné
    - pozor při přidávání souborů, rozsáhlých a/nebo významných změnách
    - úplné sestavení: dělat na čistém (novém) workspace
  - vynechat postupy pro balení, vkládání info o verzi
  - pomoci si sdílením odvozených prvků (shared version cache)

[At least] make sure that everyone compiles the same way, using the same tools, against the same set of dependencies.

Integrační sestavení

Microsoft Windows NT 3.0 consisted of 5.6 million lines of code spread across 40,000 source files. A complete build took as many as 19 hours on several machines, but the NT development team still managed to build every day.
– McConnell, 1996

- Cíl: spolehlivě ověřit, že produkt jde sestavit
  - soukromý build nestačí
    - složité závislosti, špecifiká ve workspace, zjednodušení pro zrychlení
  - úplné sestavení trvá dlouho => nemůže provádět vývojář

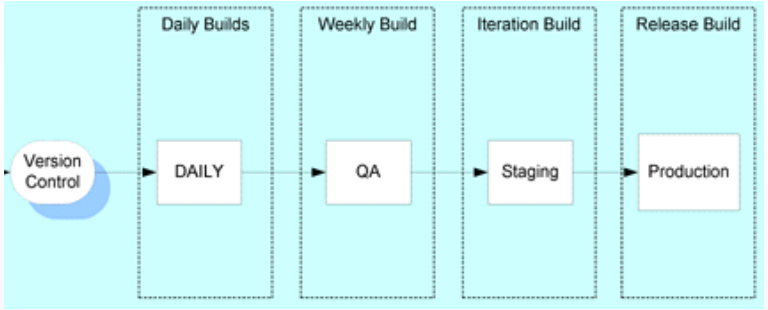
- Postup: celý produkt (vč. závislostí) sestaven centrálně, automatizovaným a opakovatelným procesem
  - postup co nejpodobnější sestavení pro release
    - vždy „na zelené louce“ (clean full build)
  - maximální automatizace - typicky běží přes noc
  - spolehlivé mechanismy zaznamenání chyb a informování o nich
    - emailové notifikace začátek, konec, výsledek
    - web s přehledy a detaily
  - úspěšné sestavení může být označkováno ve verzovacím systému

Kusovník, archivace prostředí

- Kusovník
  - kompletní seznam prvků sestavení
    - reprodukovatelnost sestavení kdekoli, kdykoli
    - zejména při distribuovaném nebo jinak složitém buildu
  - samoidentifikující konfigurace pomůže
    - znalost verzí bez přístupu k verzovacímu systému
  - viz strojírenství ; automatizace (ClearCase make)

- Archivace prostředí
  - správa verzí objektů, které nejsou v úložišti
    - nástroje, platformy, hardware, prostředí - identifikovat sestavení
  - klíčové pro dlouho žijící software (např. povinné v letectví)

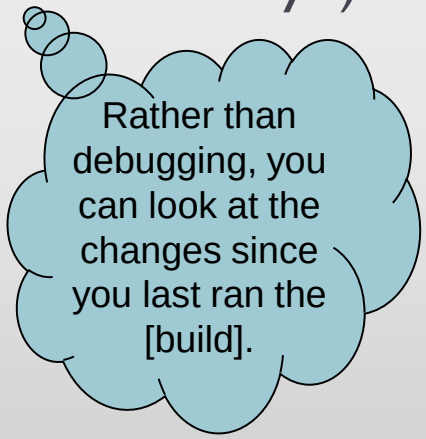
Release build



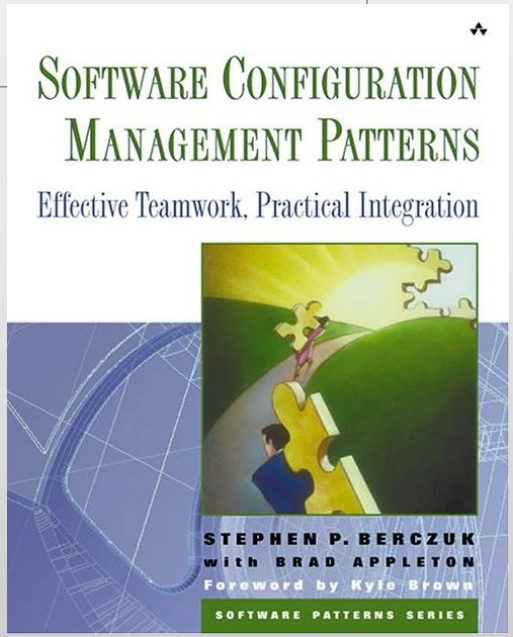
- Význačné integrační sestavení: dodáno zákazníkovi
  - může být interní zákazník, např. QA
- Náležitosti release
  - revize/verze konfigurace použité pro sestavení
    - které prvky, v jakých verzích (vč. 3rd party)
  - datum vytvoření
  - identifikátor sestavení
  - další metadata
    - zodpovědná osoba
    - zdrojová značka konfigurace (z verzovacího systému)
    - jakými prošlo testy, výsledky testů
    - cesta k logům překladu, testů
  - „marketingová verze“ – např. OpenCms 7.5

Diskuse o sestavení

- Automatizace, autamotizace, autmoatizace, atumoazitace, tmozaicetaua
  - celý proces (jinak bezpředmětné)
  - plánovač spouštění buildu
  - vytváření čísel/identifikátorů sestavení
  - ukládání metadat do databáze a do verzování
    - vč. výsledků testů, logů, identifikace sestavení
- Frekvence integračního sestavení
  - čím častěji tím lépe (pro menší „delta“ je snazší nalezení chyb)
  - kompromis trvání buildu
    x frekvence změn x velikost změn
- Samotné sestavení nestačí
  - viz vzory pro testování dále



Nejlepší praktiky: SCM+QA



Základní shrnutí QA

- Quality Assurance
  - dobrý a dodržovaný proces, mj. dodržování standardů
  - kontroly (review)
  - statické ověřování
  - testování
  - metriky

- Verifikace (bezchybný produkt)
  a validace (správný produkt)

Viz předmět KIV/OKS

- Viz např: http://www.amazon.com/Software-Testing-2nd-Edition-Patton/dp/0672327988

Testování

- Ověření správné funkčnosti implementace
- Úrovně testování
  - jednotkové → integrační → funkční → systémové
  - zátěžové, výkonnostní, bezpečnostní, instalační, použitelnosti
  - interní, akceptační
- Techniky pro testování
  - white-box, black-box
  - výběr dat (hraniční podmínky, fuzz testing, …)
- Automatizace: viz vztah k SCM/sestavení

Statická kontrola kódu

- Ověření formální správnosti
  + dodržování pravidel + metriky

> *Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.*
>
> ~Martin Golding

- Nástroje
  - překladač a jeho hlášení (!)
  - C: lint
  - Java: pmd, findbugs, checkstyle, JSR 305
- Postupy
  - Programming by Contract
  - review, párové programování
  - automatický build – výběr pravidel, průběžné úpravy

- Viz např: J.Kiml: Java code defect analysis – KIV/TSI podzim 2010

Zkouška těsnosti (Smoke Test)

- Cíl: ověřit, že sestavení vytvořilo funkční produkt
  - samotný překlad/linkování toto nezaručuje
  - kompletní testování trvá dlouho
    - potřebujeme rychlý základní test, jehož provedení „nebolí“
- Postup: vytvořit testy ověřující základní funkčnost, bez nároku na kompletní otestování
  - odchytí nejkřiklavější chyby, odpustí drobnosti
  - automatizace - spuštění, vyhodnocení
  - spuštění při každém buildu (vč. soukromého)
    - podle toho náročnost, rozsah, počet testů
    - může být založena na testech modulů
    - přidání nových funkcí/vlastností => nové testy těsnosti

re·gress Pronunciation Key (rĭ-grĕs′)
v. re·gressed, re·gress·ing, re·gress·es
v. intr.

1. To go back; move backward.
2. To return to a previous, usually worse or less developed state.

– dictionary.reference.com

Regresní testy

- Cíl: zajistit, aby nové funkce a vylepšení nesnižovaly kvalitu již hotového kódu
  - prevence stárnutí kódu: zanášení chyb při implementaci vylepšení
  - vymyslet mechanismus jak vytvářet vhodné testy
- Postup: ověřit build produktu pomocí testů, kterými již dříve prošel (kromě testů nových funkcí)
  - indikátor existence systémových problémů
    - určení zdroje chyby není nutné => testy integrační, modulů
  - testování změn v (nečekaných) integračních aspektech
  - spuštění při integračním sestavení, před velkými změnami
- Dobrý zdroj testů: chyby objevené QA, při validaci, zákazníkem
  - produkt selhal → napsat test, který to dokáže → přidat jej do sady regresních testů (odebírání testů ze sady není dobrý trik)

Měření [kvality] produktu

- bugtracker
- statsvn

- pokrytí testy – kódu, požadavků
- charakteristiky defektů – hustota, výskyt
- kvalita zdrojového kódu

- junit => cobertura
  - zasévání chyb

Coverage Report - org.jaxen.function

Package	# Classes	Line Coverage	Branch Coverage	Complexity
org.jaxen.function	27	64%	76%	5.373
org.jaxen.function.ext	6	63%	72%	4.235
org.jaxen.function.xslt	1	86%	100%	2.5

Classes in this Package	Line Coverage	Branch Coverage	Complexity
BooleanFunction	84%	89%	8
CeilingFunction	17%	0%	2.5
ConcatFunction	89%	100%	3
ContainsFunction	14%	0%	2.5
CountFunction	78%	100%	5
FalseFunction	20%	0%	2.5
FloorFunction	17%	0%	2.5
IdFunction	5%	0%	5.5
LangFunction	80%	100%	5.25
LastFunction	20%	0%	2.5
LocalNameFunction	73%	100%	12.5
NameFunction	65%	82%	12.5
NamespaceUriFunction	31%	36%	12.5
NormalizeSpaceFunction	95%	100%	4.5
NotFunction	20%	0%	2.5

```
110 128          else if ( nav.isElement( first ) )
111              {
112 100                  return nav.getElementQName( first );
113              }
114  28          else if ( nav.isAttribute( first ) )
115              {
116   0                  return nav.getAttributeQName( first );
117              }
118  28          else if ( nav.isProcessingInstruction( first ) )
119              {
120   0                  return nav.getProcessingInstructionTarget( first );
121              }
122  28          else if ( nav.isNamespace( first ) )
123              {
```

Daily Build and Smoke test

- Integrační sestavení + zkouška těsnosti
  - pravidelně 1x denně (někdy nočně)
  - výsledky okamžitě známy a reflektovány
    - nová hlášení problémů
    - opravy ihned zapracovány do kódu
  - check-in kódu, který vede k chybám, je neslušné chování
    - lehká (nebo i vážnější) sankce vhodná
- Výhody
  - malé množství změn během denních check-in

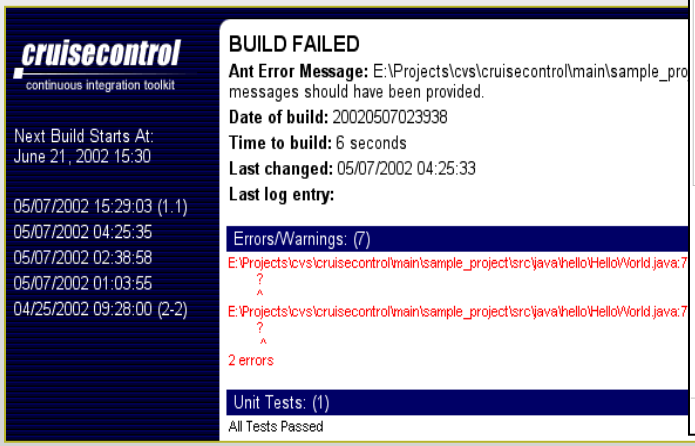
  => zvladatelné množství oprav, včas detekce problémů „vždyť včera to fungovalo“, analýza změn kódu místo ladění – viz diskuse o sestavení
  - pravidelný, obecně známý rytmus projektu
  - lepší morálka týmu („to nám to roste“)
- Cena: trocha discipliny, trocha automatizace

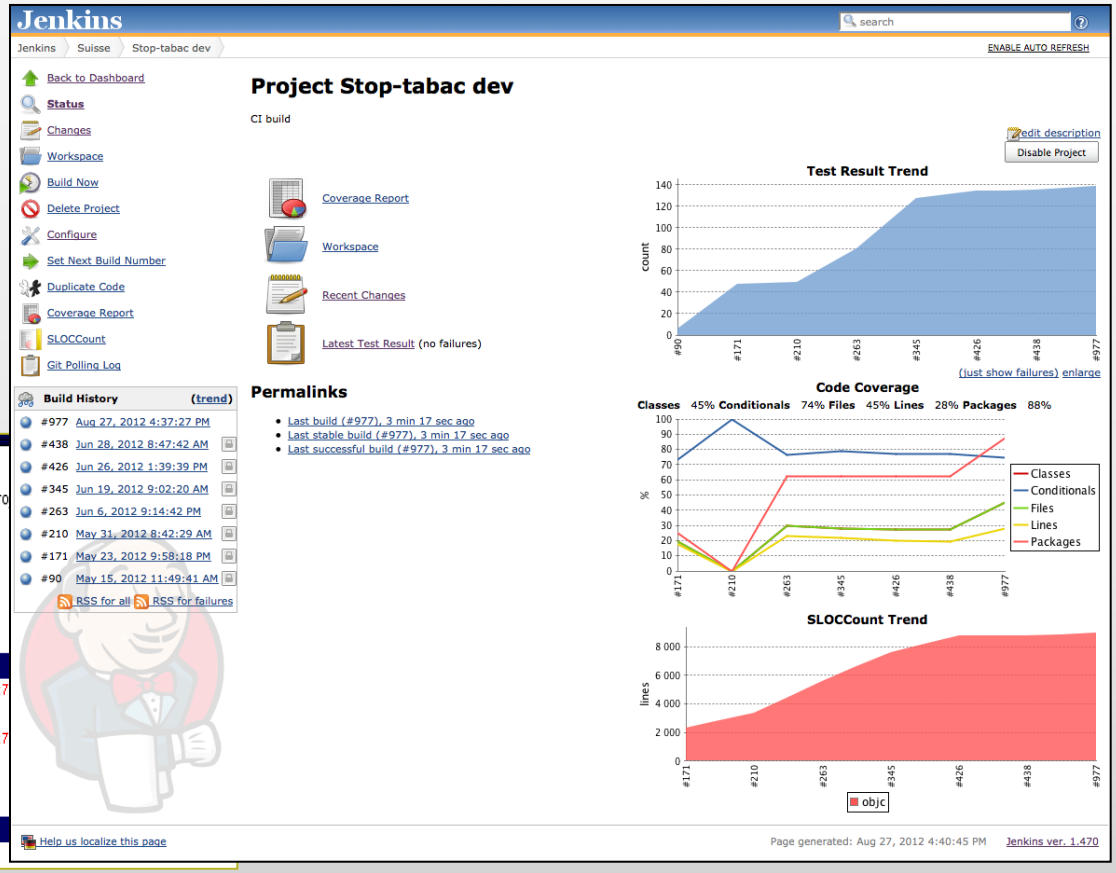
Far from being a nuisance, the [Windows] NT team attributed much of its success on that huge project [16MLOC] to their daily builds. Those of us who work on projects of less staggering proportions will have a hard time explaining why we aren't also reaping the benefits of this practice.

– Steve McConnell, 1996

Continuous Integration

- Dotažení do dokonalosti (nebo extrému ;-)
- „1x denně“ → „neustále“
- Klíč: automatizace
  - co/ci, sestavení, testování, oznamování
  - robot na spouštění





Nástroje pro podporu sestavení

- Scriptovací jazyky
  - shell, perl, python, php, …
- Buildovací nástroje
  - make
  - ant, maven, gradle
- Verifikace
  - xUnit (JUnit apod.), Selenium, …
  - testovací roboti
- Proces sestavení
  - Hudson/Jenkins, CruiseControl, …

make: příklad závislostí mezi prvky

- Build (překlad a sestavení) projektu na základě popisu závislostí typu zdrojový-odvozený
- Pojmy:
  - pravidlo (rule)
  - cíl (target)
  - závislost (dependency)
  - příkaz (command)
- Makefile: definice pravidel
  - deklarace závislostí
  - příkazy pro překlad

```
CC = gcc    # pro DOS: tcc
OBJ = main.o soubory.o

all: program
program: $(OBJ)
  $(CC) -o $@ $(OBJ)
main.o: main.c main.h soubory.h
  $(CC) -c main.c
soubory.o: soubory.c soubory.h
  $(CC) -c soubory.c

clean:
  rm *.o core
```

Maven

- Deklarativní build
- Popis struktury projektu
  - „project object model“
  - komponenty, závislosti (knihovny, jiné projekty)
  - pluginy
- Build „automaticky“
  - pro většinu jazyků a typů projektů předdefinované tasky => postup
  - získání artefaktů závislostí: úložiště centrální + lokální

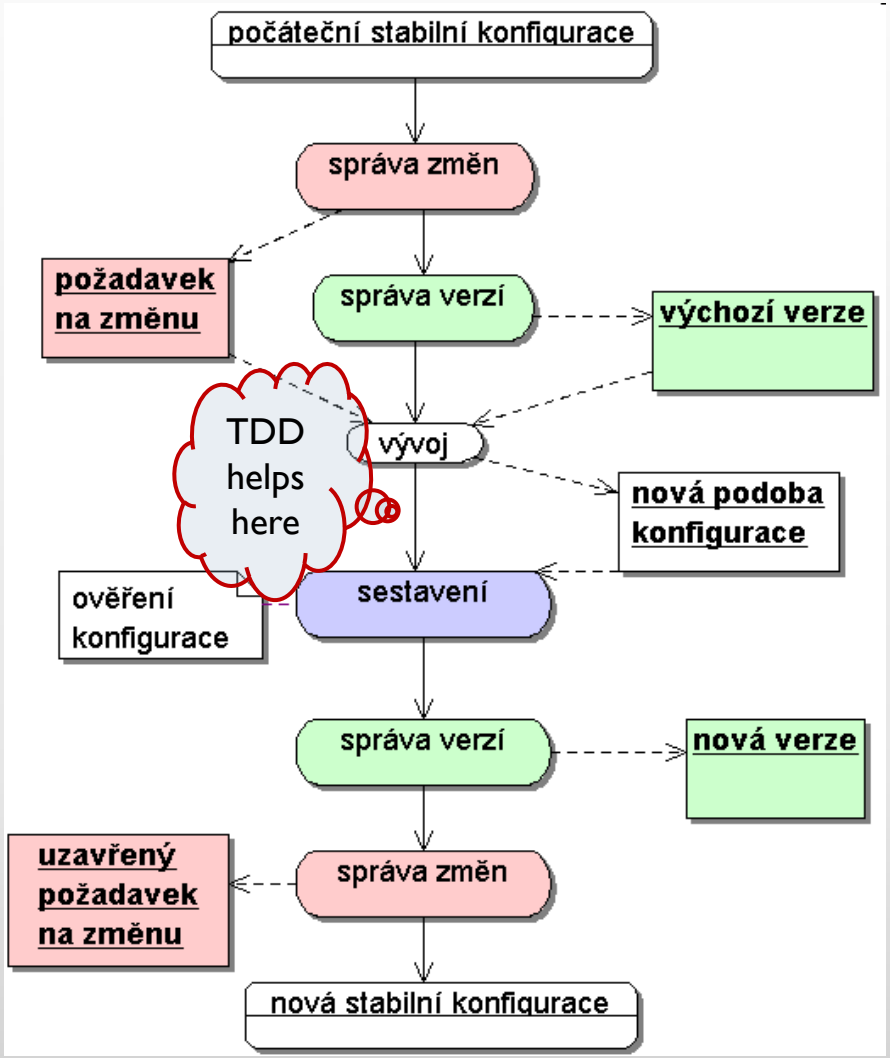

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/mav

    <modelVersion>4.0.0</modelVersion>
    <groupId>cz.zcu.kiv.spot</groupId>
    <artifactId>spot</artifactId>
    <packaging>war</packaging>

    <version>2.0-rc9</version>
    <name>spot</name>
    <description>Odborny prekladovy slovnik odborne terminologie.</description>

    <properties>
        <spring.version>2.5.6</spring.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.4</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- dalsich N vynechano ... -->
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.mortbay.jetty</groupId>
                <artifactId>maven-jetty-plugin</artifactId>
                <version>6.1.10</version>
                <configuration>
                    <scanIntervalSeconds>10</scanIntervalSeconds>
```

Hudson / Jenkins / Travis

- Automatický build a průběžná integrace
  - vyhrazený stroj
  - website
- Spuštění buildu
  - post-commit hook, VCS polling
  - upstream project, ručně
- Konfigurace buildu
  - závisí na vlastním build nástroji (ant, maven, …)
- Informace
  - dashboard projektu, statistiky
  - drilldown (důvody neúspěchu, logy, konzole)

Shrnutí

SCM = základní hygienické návyky



9:12 ranní káva

9:30 daily standup (a debata o Checkstyle)

9:55 kontrola emailů z nočního buildu (OK)

10:03 kontrola práce v bugzille, #10231 už dělá KJA, beru si *#11209 (assign to self)*

10:10 *svn update*, nutný merge na úpravy ze včera ale bylo to za 10 minut hotovo

10:48 bugfix byl rychlý, teď *mvn smoke-test* → Test FAILed! … ach jo

11:12 smoke test konečně OK, pošleme to do cukrovaru…

11:13 *svn update* (naštěstí jsem nejrychlejší, žádný konflikt :)

11:13 *svn ci -m „Oprava bug #11209 Nefunkční ukládání velkých příloh“* .

11:14 email od bugzilla: *User PBA commented on #11209 (revision 34221)*

11:30 email z buildu: *revision 34221 builds OK* – krása, jde se na oběd!

11:32 skleróza: *bugzilla #11209 status: test*