

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 2017/2018

► 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



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

Samotná výroba není až tak složitá. Evropská vesmírná agentura má ale velmi vysoké nároky na kvalitu, přesnost a stabilitu produkce. Dostat se na trh a získat certifikaci je proto nesmírně náročné. "Konkrétní hotový díl ještě není sám o sobě výsledkem. Musí mít od začátku jasnou identifikaci. Přesně víme, z jaké šarže pochází, jaký materiál byl použit, kdy jsme práci provedli a co konkrétně vzniklo,"

*-- Jan Bureš, ředitel výroby Aerotech
Czech. iHNed, 1.2.2017*

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

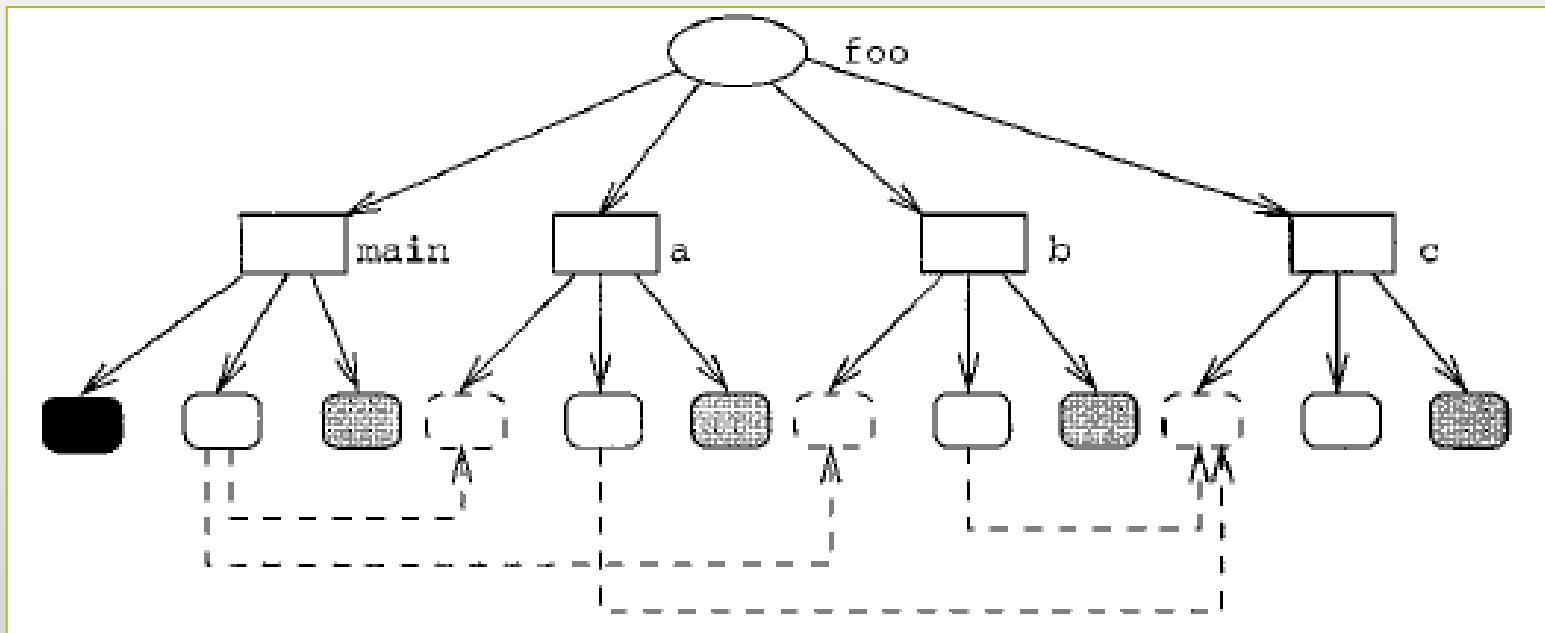
- Prvek konfigurace = konstituující složka systému
 - různé typy: zdrojový soubor, dokument, model, knihovna, script, spustitelný soubor, testovací data, ...
- **Ve správě SCM**
 - => ví se o jeho existenci, vlastníkově, změnách, umístění v produktu
 - **atomický** z hlediska identifikace, změn
 - jednoznačně 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

► Vztahy v konfiguraci

- celek-část, master-dependent
 - určují strukturu a závislosti
- zdrojový-odvozený
 - určují způsob produkce, tj. build produktu



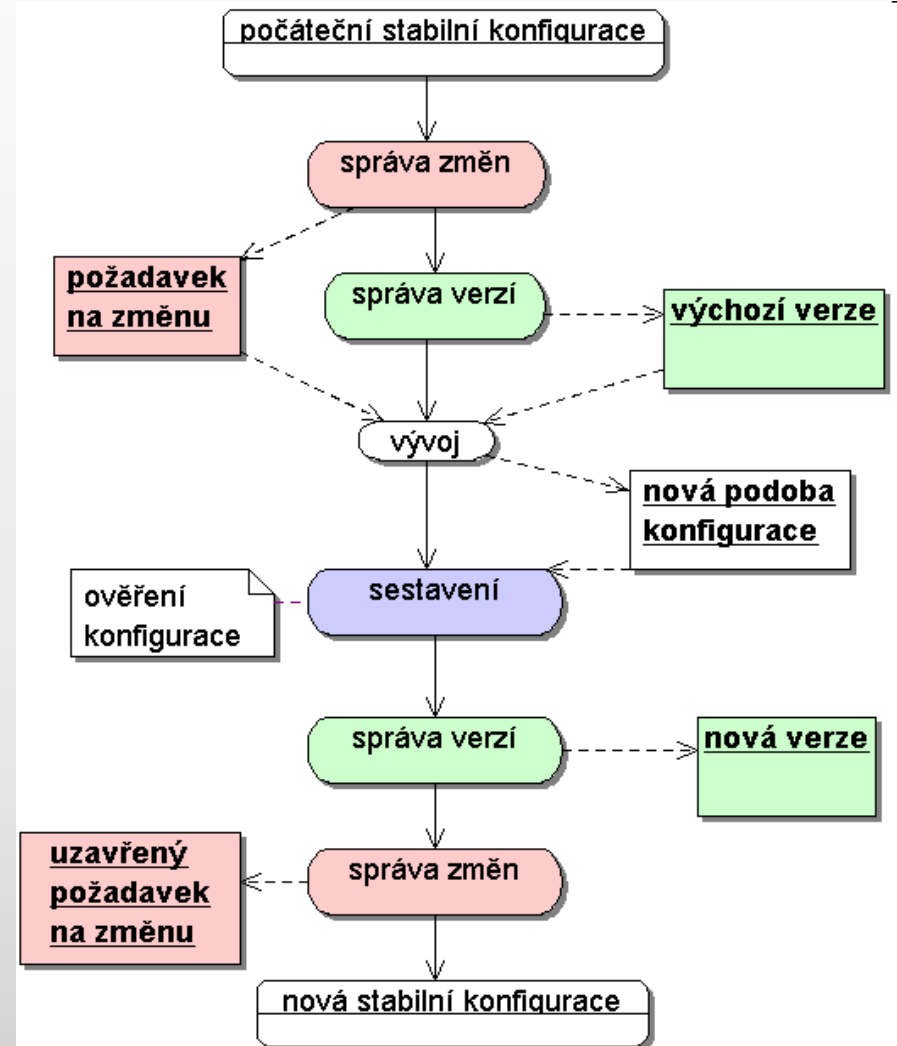
► Úlohy SCM

- 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

► Aktivita SCM v cyklu vývoje



- Správa změn
- Identifikace a správa verzí
- Sestavení



SCM: správa změn

Change is inevitable
except from vending machines.



CHANGE
IS
HAPPENING

► 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 \leftrightarrow 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*

► Proces správy změn

► Požadavek na změnu – stavy

- vytvořený
- schválený
- přiřazený
- vyřešený
- ověřený
- uzavřený
- znovuotevřený
- odmítnutý

► Během provádění

- znovuotevření problémů
- vygenerování nových hlášení



► Ticket – detaily (I)



► Při **vytvoření**

- nutné náležitosti: shrnutí (+ id, autor, datum)
- **popis**, co nejpřesnější
 - jak chyba vznikla
 - jak reprodukovat
- screenshot, **vzorek dat**, ...

► *dress code: informal × formal*

Příklad – dobrý a špatný bugreport

- **závažnost**, priorita (odhad)
- **konfigurace** daného sw a systému (OS, knihovny, ...)
 - včetně identifikace verzí

► Ticket – 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



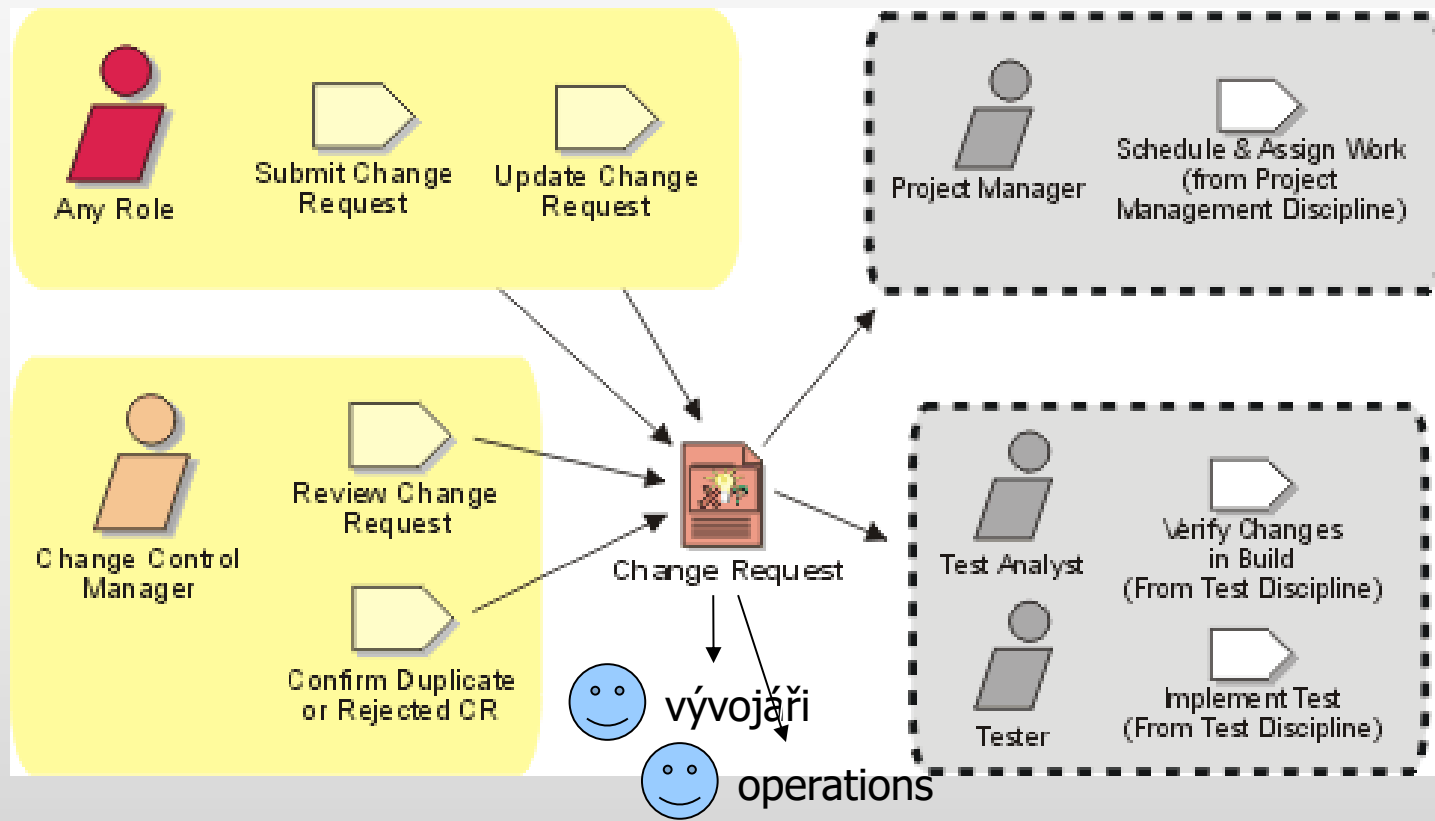
► Akce při zpracování ticketu

- 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



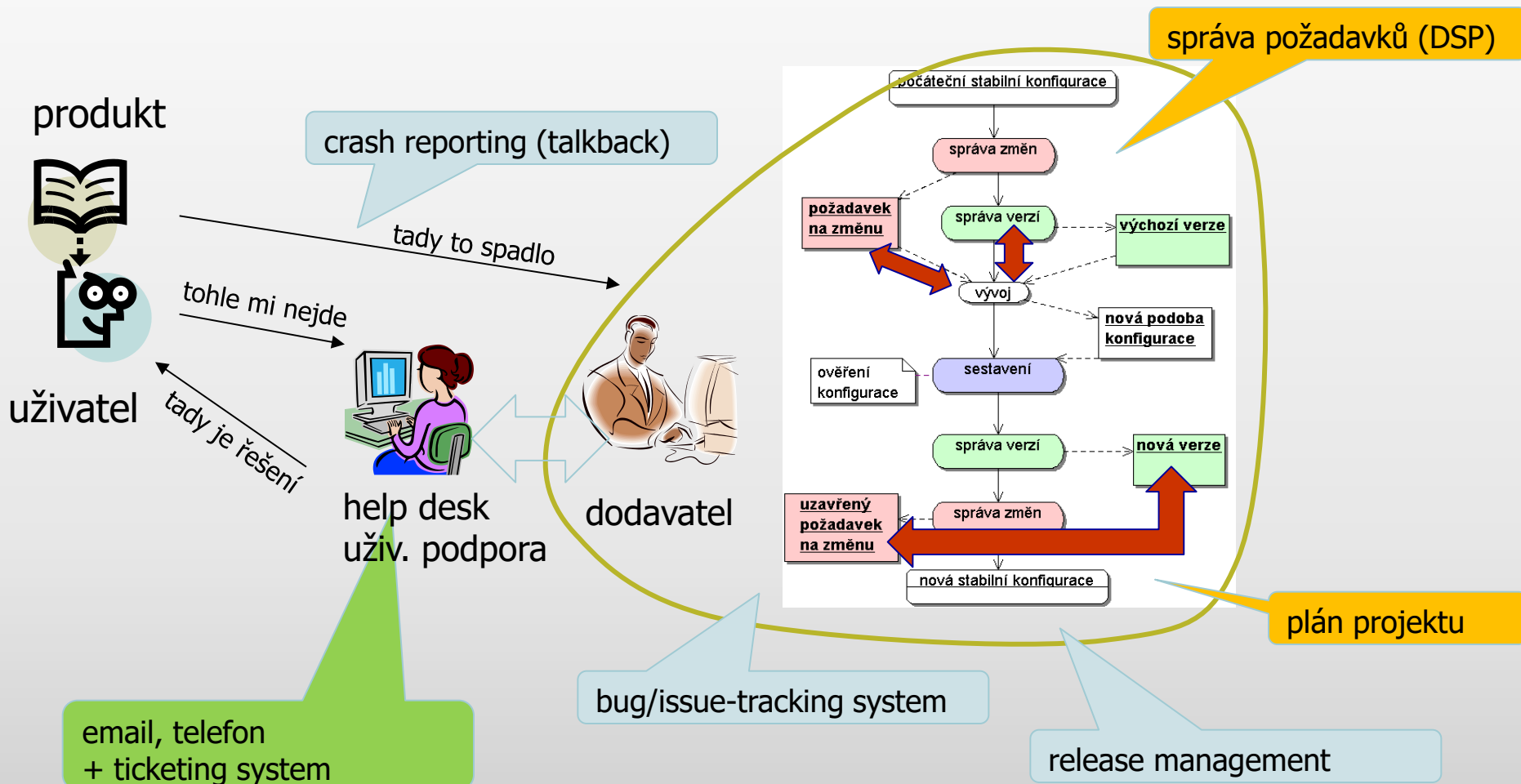
► Change Control Board

- 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



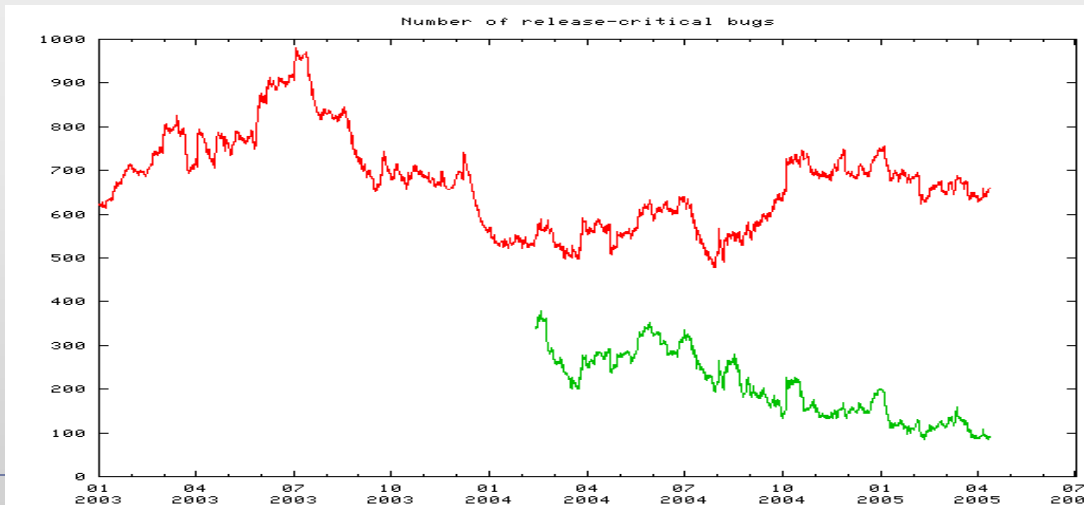
► Správa změn a řízení projektu

► Kritéria dodávky (**release management**)

- časový termín
- implementovaná funkčnost / vlastnosti
- kvalita produktu

Viz backlog,
burndown

- Jak určit termín podle kvality? Konverguje nám iterace?
- **Změnové řízení je jádrem řízení ve fázi údržby**



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

► 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 = ticket typu feature request

► Workflow

- vize
- požadavky □ DSP
- feature requests □ bugtracker
- (testy)
- bug report a/nebo re-open feature request



Backlog spojuje
toto vše do
jedné struktury

► Flexibilita, souvislost s plánováním



► 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

Main | View Bugs | Report Bug | Summary | Docs | My Account | Logout

Reporter: any Assigned To: any Category: bugtracker Severity: any Status: any 50

Search:

any
feature
trivial
text
tweak
minor
major
crash
block

Viewing Bugs (1 - 50 / 416) [Print Reports] [CSV Export]

	P	ID	#	Category	Severity	Status	Updated	Summary
		0003155	1	bugtracker	minor	resolved (vbactor)	04-23-03	Field 'date_added' not correctly printed in print_bug_page.php
		0003154	1	bugtracker	minor	new	04-23-03	The attached documents are not sorted
		0003153	1	bugtracker	feature	resolved (vbactor)	04-23-03	Change the word 'bug' to 'issue'
		0003152	4	bugtracker	feature	feedback	04-23-03	Adding notes and documents to resolved bugs
		0003151	1	bugtracker	feature	resolved (vbactor)	04-23-03	Change alignment of input fields in forms
		0003150		bugtracker	feature	new	04-22-03	More flexible CSV export
		0003149	1	bugtracker	feature	new	04-22-03	"Resolve bugs" needs more available prompts
		0003147		bugtracker	major	new	04-22-03	unable to attach files to bugs.
		0003144		bugtracker	minor	new	04-21-03	Can't sort in proj_user_menu_page.php
		0003143	2	bugtracker	tweak	resolved (vbactor)	04-23-03	Show a padlock instead of "p"

http://bugs.flyspray.org/proj1?do=index

FLYSPRAY

Premek Brada (obrada) | My searches | Logout

Flyspray - The bug killer! | Switch | Overview | Tasklist | Add new task | Roadmap

This is the Bug Tracking System for the Flyspray project. **This is not a demo!** Before opening a new task, please read the guidelines!

Do not issue bugs reports against versions earlier than 0.9.9.1

Security problem? Check the security section.

Search this project for: Advanced Save search as: OK

ID	Task Type	Summary	Status	Progress	Severity	Votes
1222	Feature Request	Wanted - a workflow engine / Role-based State Transi...	Unconfirmed	2	Medium	0
1174	TODO	TODO List for 1.0	New	5	Medium	1
407	Feature Request	Plugin system	Assigned	22	Medium	2
1226	Feature Request	Change language with GET parameter	Unconfirmed	1	Low	0
1214	Feature Request	Put all files in subdirectory in distribution zipfile	Assigned	5	Low	0
1195	Bug Report	Incorrect rendering in IE 6.0 and 7.0	Unconfirmed	11	Low	0
1158	Feature Request	SVN/CVS integration	Assigned	4	Low	0
1134	Feature Request	add icon/image for each project	Unconfirmed	3	Low	0
1090	Bug Report	Hide some fields and add a javascript link to show thes...	Unconfirmed	4	Low	0
999	Feature Request	define each role when assign a task for several users	Unconfirmed	2	Low	2
961	Feature Request	Email bug reporting	Unconfirmed	14	Low	2
937	TODO	New fancy layout for 1.0	New	3	Low	0

► ALM: Redmine, Jira+Confluence, RTC

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

The screenshot displays the Redmine web application interface. At the top, there is a navigation bar with links: Úvodní, Projekty, Návodě, Přihlášení, and Registrovat. Below this is a search bar labeled 'Hledat:'. The main navigation menu includes: Přehled, Download, Aktivita, Plán (selected), Úkoly, Novinky, Wiki, Fóra, and Repozitář.

The 'Plán' (Plan) section is active, showing three project plans:

- Plan 2.6.4**: 33% completion. 3 tasks (1 closed, 2 open).
 - Defect #19297: Custom fields with hidden/read-only combination displayed in Issue Edit Form
 - Defect #19400: Possibility of having 2 (or more) repositories with empty identifier
 - Defect #19444: Fix typo in wiki_syntax_detailed.html
- Plan 3.0.2**: 33% completion. 9 tasks (3 closed, 6 open).
 - Defect #19164: 500 Internal error: sorting for column mail at Administration/User
 - Defect #19163: Bulk edit form shows additional custom fields
 - Defect #19168: User activity: changes made to tickets are shown multiple times
 - Defect #19185: Update Install/Upgrade guide for 3.x version and get rid of DEPRECATION WARNING: You didn't set config.secret_key_base
 - Defect #19276: Creating new issues with invalid project_id should return 422 instead of 403 error
 - Defect #19405: Setting config.logger.level in additional_environment.rb has no effect
 - Defect #19553: When create by copying the issue, status can not be changed.
 - Defect #19558: Mail handler and x-auto-response-suppress
 - Feature #19437: Upgrade to Rails 4.2.1
- Plan 3.1.0**: 14% completion. 14 tasks (2 closed, 12 open).
 - Defect #4334: "Watch"ing an issue doesn't update watchers list
 - Defect #19162: new issue AND tracker read-only: status is not changable
 - Defect #19243: Ambiguous date format options (eg. 03/03/2015) in settings
 - Defect #19403: Mistake in Polish Translation file

On the right side, there is a sidebar with a 'Plán' section containing checkboxes for Defect, Feature, and Patch, and a 'Zobrazit dokončené verze' (Show completed versions) link. Below this is a 'Verze' (Versions) section listing 2.6.4, 3.0.2, and 3.1.0, with their respective statuses: 'Candidate for next major release', 'Candidate for next minor release', and 'Unplanned'. A 'Dokončené verze' (Completed versions) link is also present.

Příklad – vyhledávání > seznam > detail > historie



WHAT SOURCE CONTROL
DO YOU USE?



I JUST ZIP
UP THE FOLDER

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)

► Typy verzí



► Základní druhy verzí

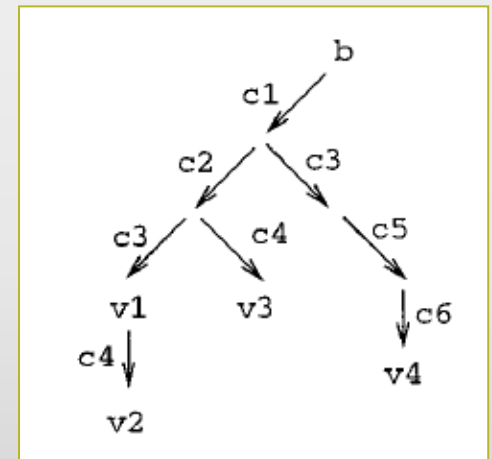
- historická podoba □ **revize** („Word 6.0“)
- alternativní podoba □ **varianta** („Word pro Mac“)

► Verzování podle **stavu**

- identifikují se pouze podoby prvků
- výsledná verze vznikne vhodným výběrem

► Verzování podle **změn**

- identifikují se (také) změny prvků (viz později **changeset**)
- výsledná verze vznikne aplikací změn



► Granularita verzování

► Product Space x Version Space

► verzování **komponent**

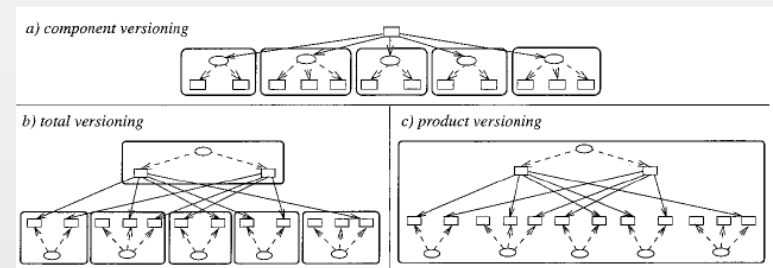
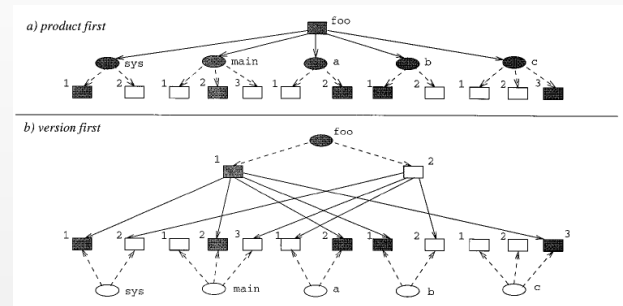
- jednotlivé prvky samostatně
- konfigurace nemá verzi

► **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



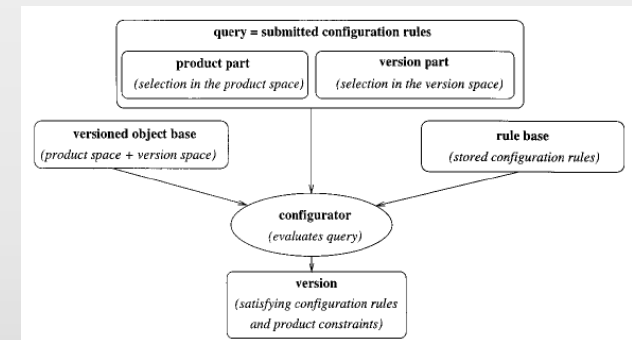
► Identifikace konkrétní verze

► 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

<http://semver.org/>

► 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

► Meta-data (dostupná z úložiště x embedded)

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

Příklad – firemní dokument,
RCS keywords

► 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 bootstrap 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 (I)

- 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, mercurial, RTC
- unikátní ID commitu, HEAD vs TIP

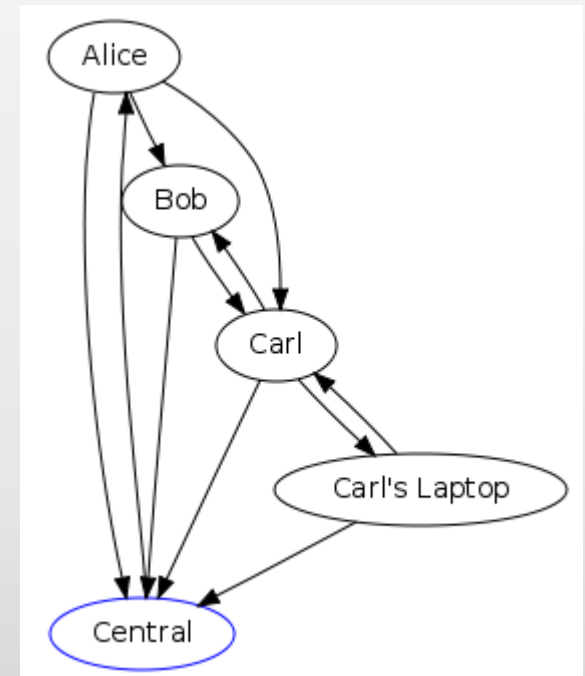


Image source: <http://mercurial.selenic.com/wiki/UnderstandingMercurial>

► 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 + GitHub“

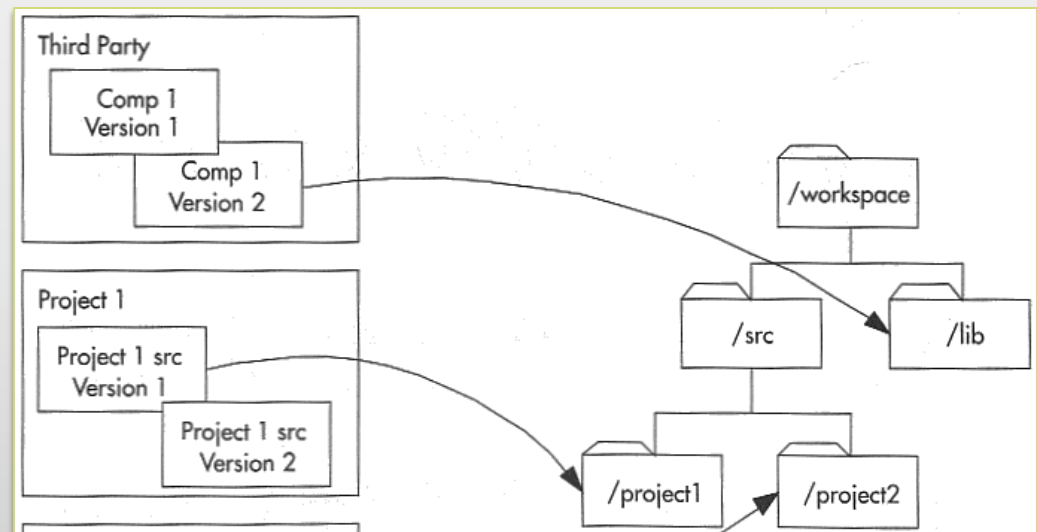
► Pracovní prostor

- 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

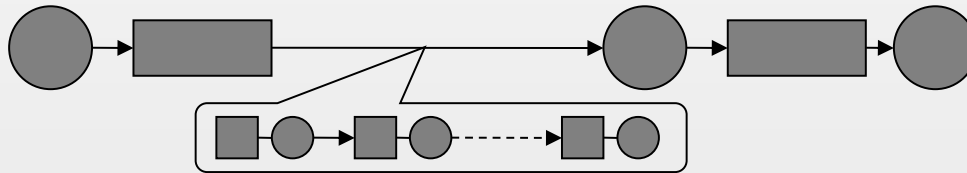
- *checkout*
- *clone*



► Codeline



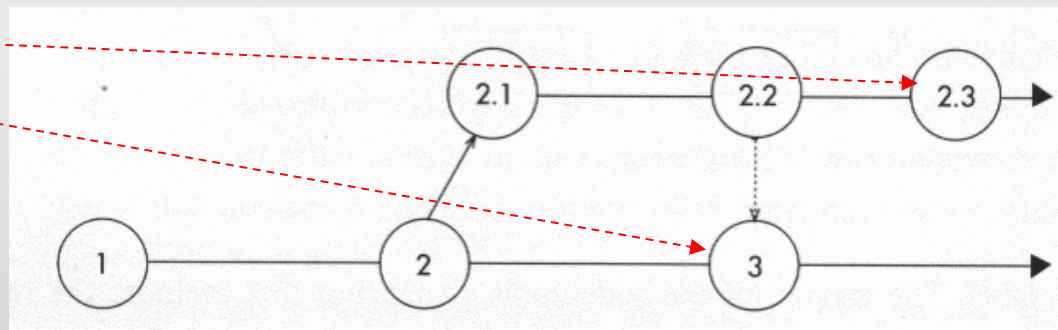
- Codeline (vývojová linie) = série podob (verzí) množiny prvků konfigurace tak, jak se mění v čase
 - (extenzionální verzování podle stavu) obsahuje **commity** a/nebo changesety, větve => část grafu verzí



- vrchol codeline obsahuje nejčerstvější verzi

- “HEAD”, “tip”

- má přiřazena **pravidla** práce na codeline



► Pravidla na codeline

► Codeline má přiřazena **pravidla práce** (policy)

- klíčová součást (viz architektura – konvence)
- odlišení různých codelines

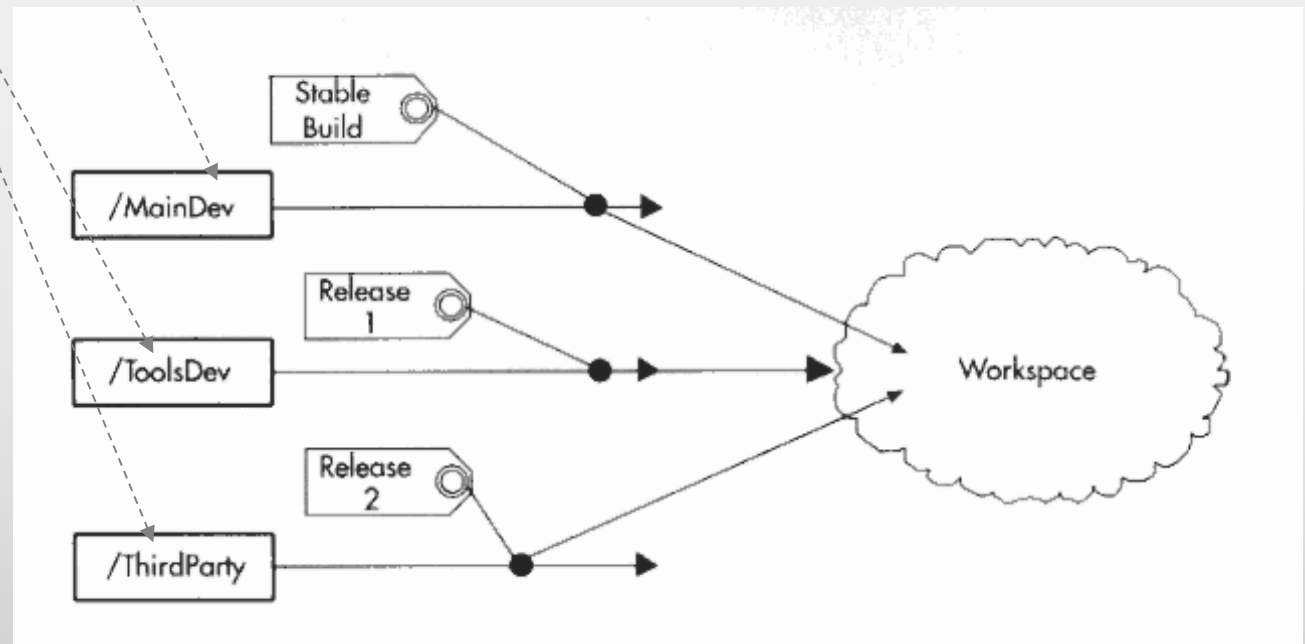
- typ prací (vývoj, údržba, release, ...)
- očekávaná kvalita kódu
- pravidla/akce před ci, po checkout, ...
- jak a kdy ci, co, branch, merge
- přístup pro osoby a skupiny
- kam (codeline) exportovat změny, odkud importovat
- doba platnosti či podmínky odstavení

Viz **A successful Git branching model**



► Codeline (2)

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



► Delta, diff a changeset

- Diff: např. ``diff file-a file-b``

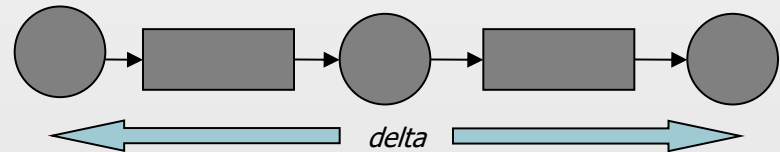
- viz též ``patch``

Příklad – RCS `*,v` + 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

- související delty tvořící atomický celek, ve správě SCM
 - obvykle též “+ důvod” tj. vazba na správu změn
 - viz **OpenUp Concept: Change Set**

► Tag, baseline

- **Tag**, label = označení konfigurace symbolickým jménem

- štítek
- různé způsoby realizace



- **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

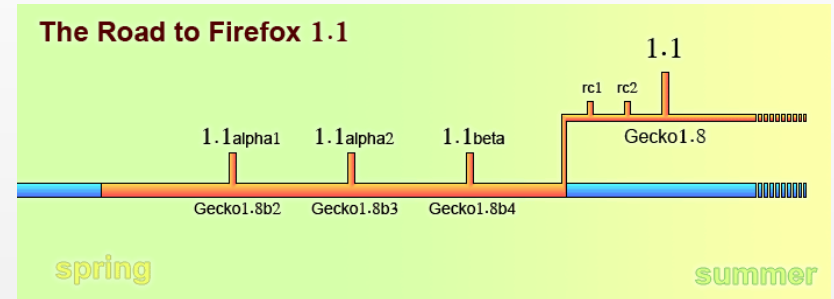
► Význačné baseline

► Interní release

- iterace

► Milníky projektu

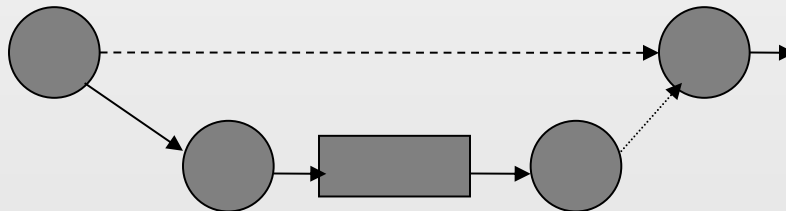
- jednotlivé fáze životního cyklu
 - vodopád / spirála / iterativní
 - alfa verze = „feature complete“, interní testování
 - beta verze = testování u (vybraných) zákazníků
- ## ► Finální release produktu
- verze dodané zákazníkovi/na trh



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

► Paralelní práce

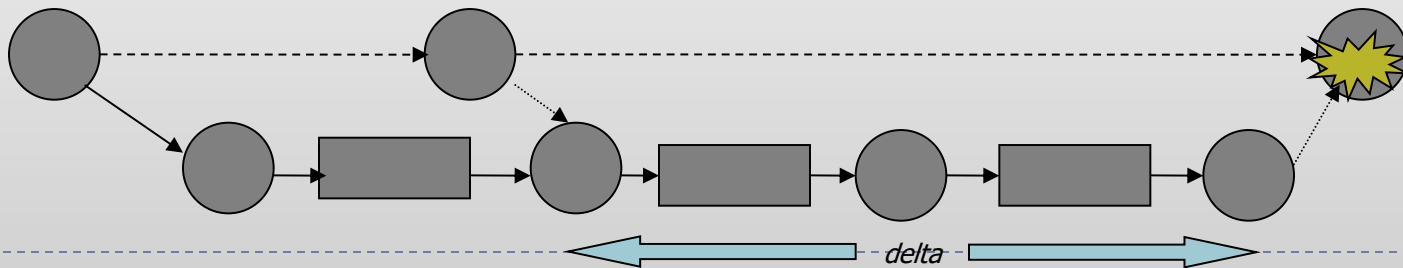
- Souběžné úpravy stejné konfigurace spravované VCS
- Důvod: velké úpravy, release, spekulativní vývoj, varianty, ...
- Cíl: vzájemná **izolace paralelních prací**
 - aby ukládané změny během nich neovlivnily ostatní
 - => oddělení paralelních vývojových linií



- 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

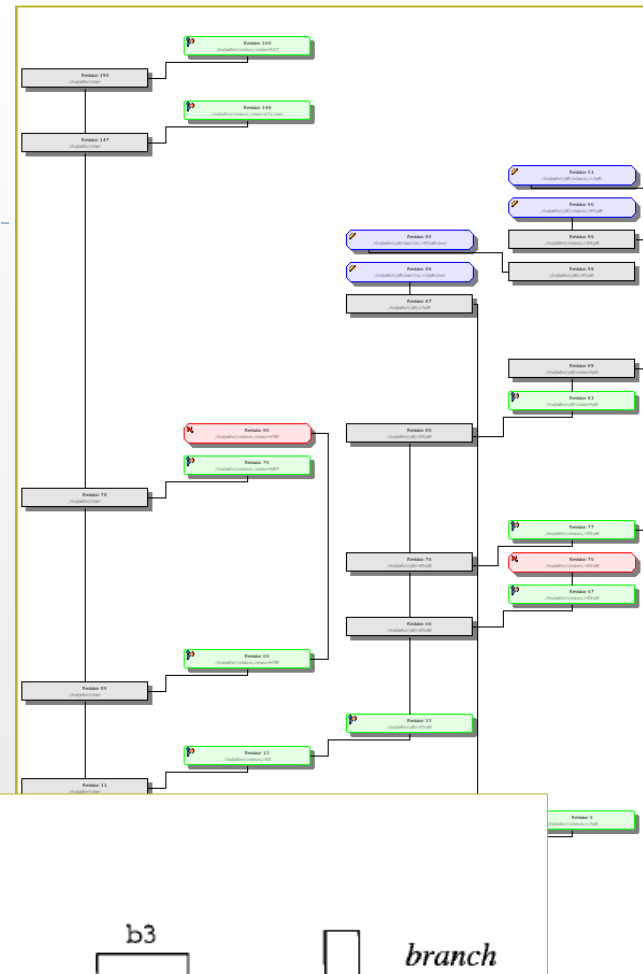
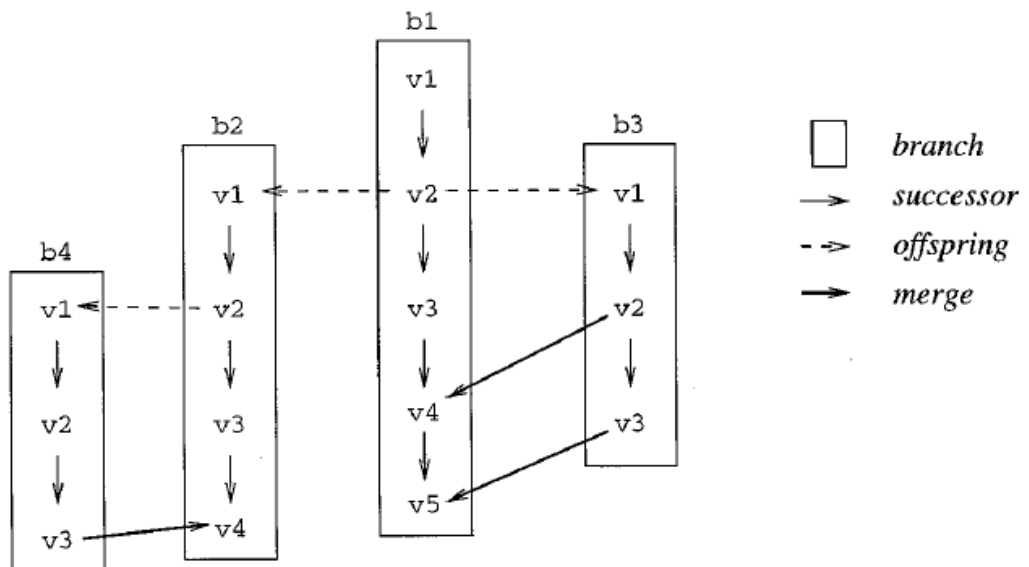


► Graf verzí

- 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, feature)  
| * d384a57 - Migrating HTML, CSS and  
| * dc47a96 - README.md (Christophe  
| * dca0784 - Boulot du collègue (Ch  
| * 80b0beb - Footer tweak + MIT lice  
| / \  
* 6d731f3 - Content tweaks (Christoph
```

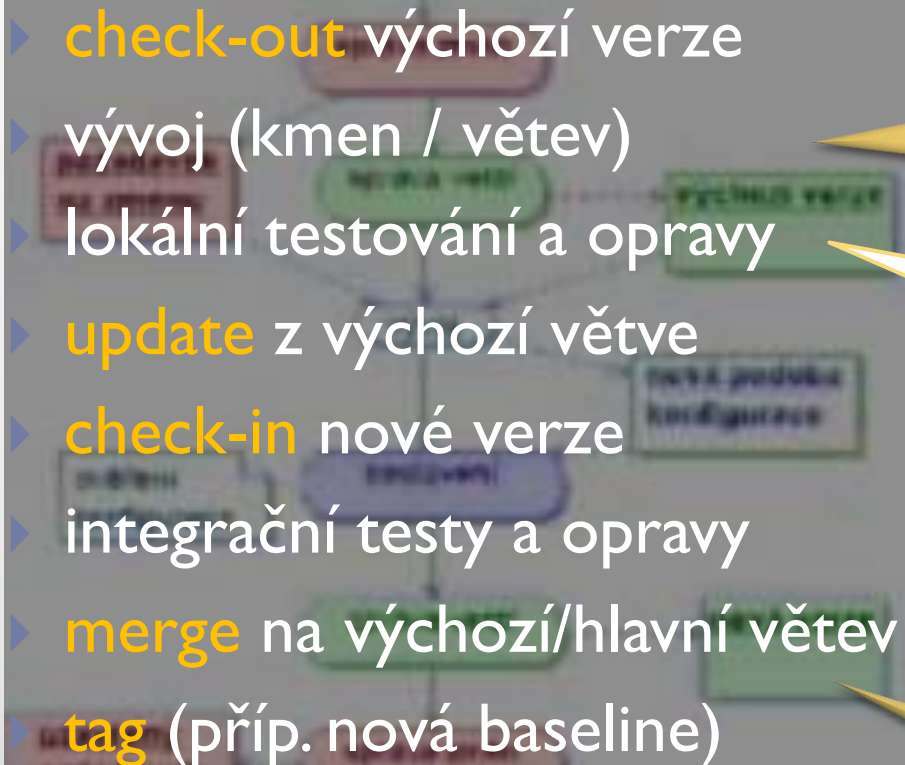
Příklad – svn / git graf





Použití správy verzí

► Verzování v procesu řešení úkolu

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

Které? Viz správa verzí:

- výchozí pro vývoj
- starší release pro bugfix

Kam? Viz správa verzí:

- private / task branch / ...

Kolik? Viz změnové řízení:

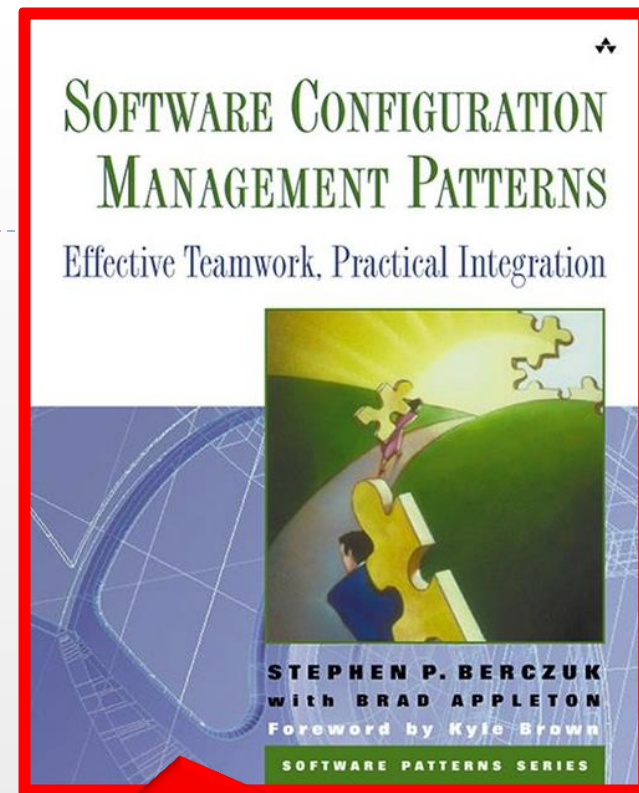
- commit per task
- commit per bug/request
- lokální commity

Kam? Viz správa verzí:

- hlavní vývojová linie
- větev (release, task)

► Vzory pro verzování (I)

- Privátní verze (*private versions*)
 - soukromé úložiště pro častější check-in
- Check-in pro úkol (*task-level commit*)
 - minimum nutného
- Větev pro úkol (*task branch*)
 - složitější úpravy s většími následky
- Hlavní vývojová linie (*mainline*) a pravidla (*codeline policy*)
 - jedna codeline pro průběžný vývoj



viz doporučené pořadí čtení

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



▶ **Vzory pro verzování (2)**

- ▶ Stabilizační období (*code freeze*)
 - ▶ pravidla před release
- ▶ Větev pro release a jeho přípravu (*release prep, rel line*)
 - ▶ místo code freeze
- ▶ Správa kódu třetích stran (*third party codeline*)
 - ▶ vlastní větev pro každý kód od externího dodavatele

► Nástroje pro verzování (VCS)

Version Control Systems

- **Ruční verzování**
 - souborový systém + pravidla
- **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, ALM**
 - 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
```

► **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)

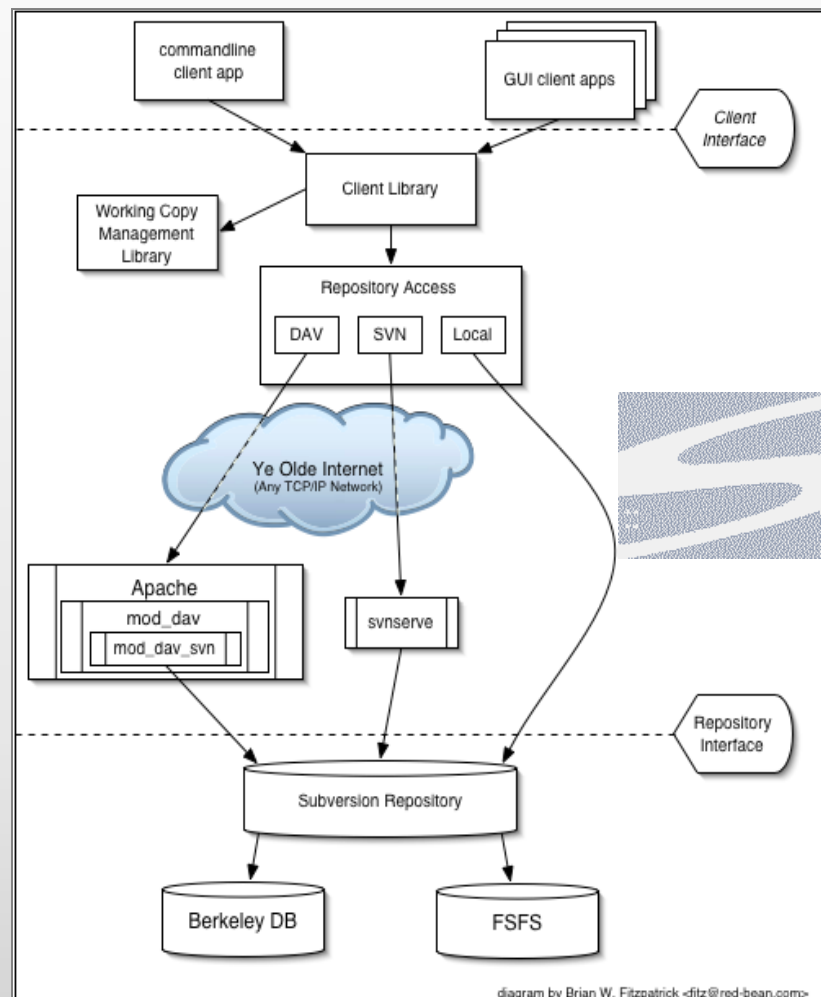
▶ CVS (Concurrent Versioning System)

- ▶ 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
 - ▶ trigger
 - ▶ 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

► 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



► svn: Několik poznámek

► Identifikace verzí

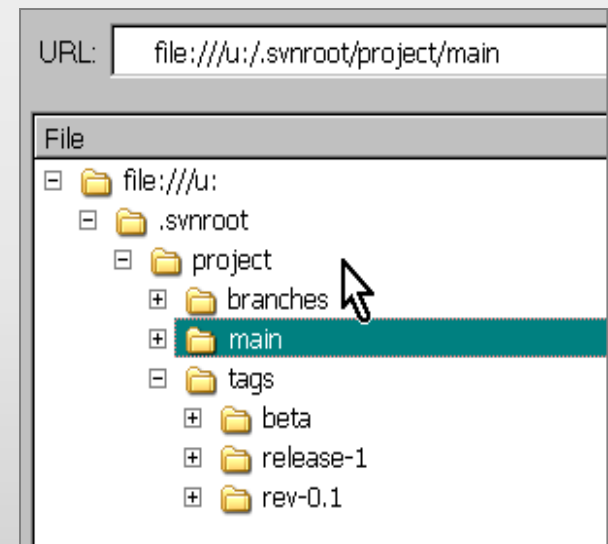
- globální kontinuální jednočíselné identifikátory (číslojí commit)
- není koncept „značek“ (tag) jako metadat

► 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



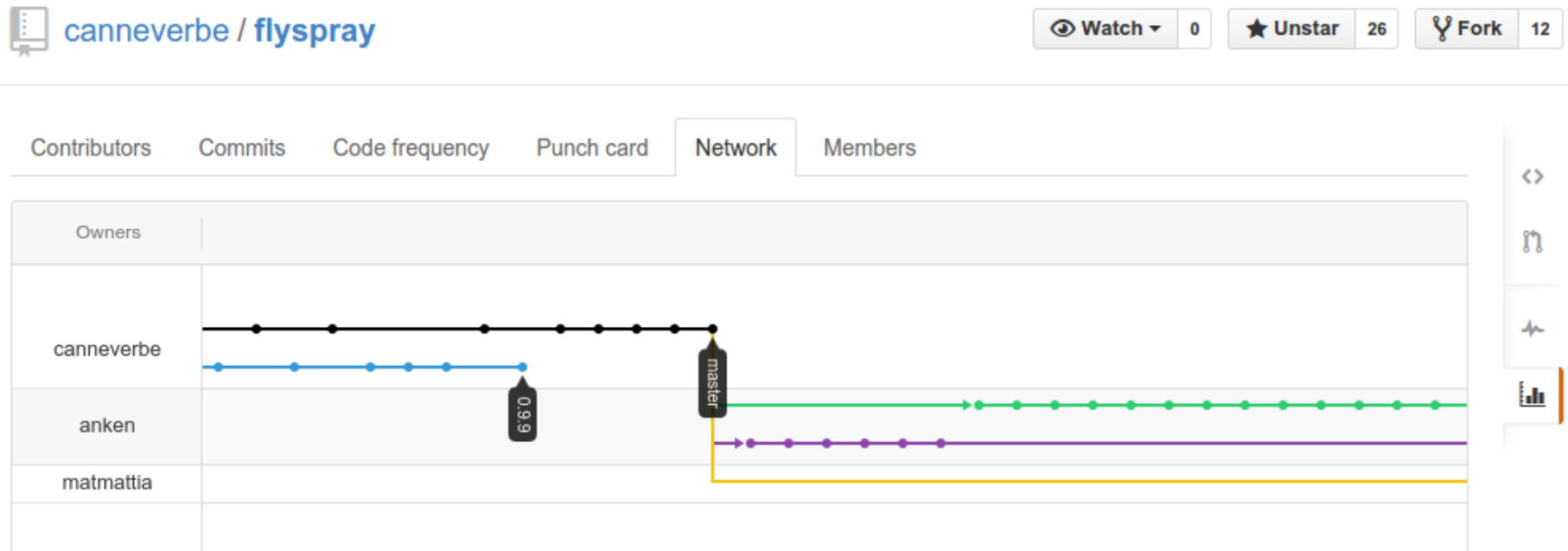
git

--distributed-is-the-new-centralized

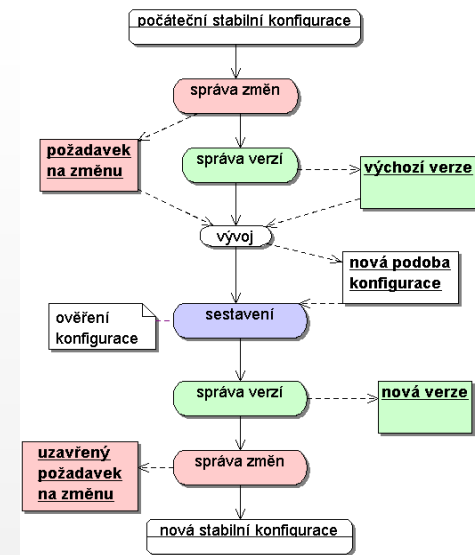
► Nadstavby Git

► GitHub (a jeho klony) – „social coding“

- velmi snadné sdílení příspěvků: repository fork, **pull request**



► Git LFS; Gerrit



SCM: řízení sestavení (build)

► Řízení sestavení

- Aktivita provádějící transformaci zdrojových prvků konfigurace na odvozené
 - zejména sestavení celého produktu
- Cíl: vytvořit systematický a automatizovaný postup
- Pojmy
 - sestavení / build (též integration) – proces a výsledek (artefakt) vytvoření částečné nebo úplné podoby aplikace
 - zdrojové a odvozené prvky konfigurace



► Sestavení: proces

► Build process

- míra formálnosti
- míra preciznosti

► Kroky

1. příprava
2. check-out
3. preprocessing
4. překlad, linkování
5. nasazení
6. spuštění
7. testování
8. značkování, check-in
9. informování

Sestavení != `gcc a.c`



► 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/
```



► Sestavení: pravidla

► **Jedinečnost** a identifikovatelnost

- sestavení jako artefakt: 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



▶ **Nástroje pro podporu sestavení**

▶ **Scriptovací a preprocessing jazyky**

- ▶ cpp, m4, ...
- ▶ shell, perl, python, php, ...

▶ **Buildovací nástroje**

- ▶ make
- ▶ ant, maven, gradle

▶ **Verifikace**

- ▶ xUnit (JUnit apod.), Selenium, ...
- ▶ testovací roboti, virtualizační a kontejnerové systémy

▶ **Řízení procesu 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



▶ 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)

Základní typy a vzory pro sestavení

► Typy sestavení



► Obsah sestavení (co je použito)

- čas překladu x jistota správnosti
- přírůstkový (inkrementální) build
- čistý
- úplný

► Účel sestavení

- jak standardizované je prostředí
- soukromý
- integrační (oficiální)
- release build

► Vzory pro sestavení (I)

► Základní postupy

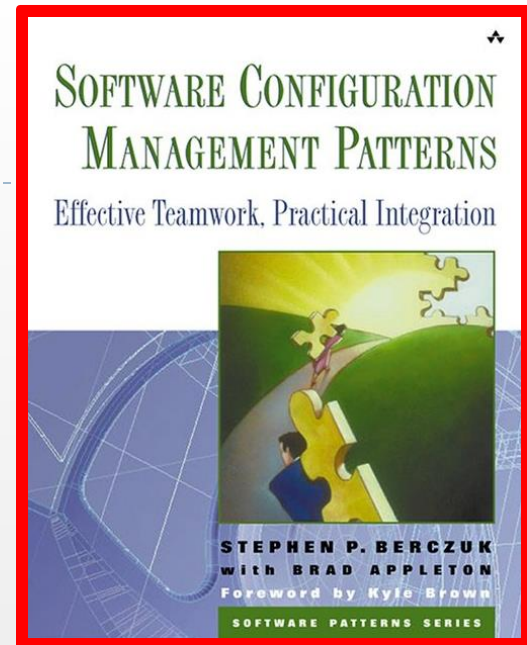
- soukromé sestavení (*private system build*)
+ sdílení součástí
- integrační sestavení (*integration build*)

► QA-related postupy

- zkouška těsnosti (*smoke test*)
- regresní testy (*regression test*)

► Obecný cíl:

odchytit co nejdříve okamžik kdy „se to rozbilo“



► Vzory pro sestavení (2)

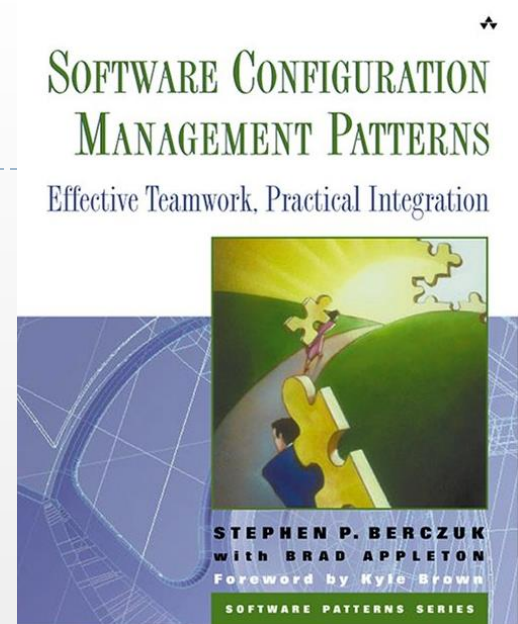
► Vydávání produktu

- release build
- balení a distribuce (packaging)

► Podpůrné aktivity

- kusovník (Bill of Materials)
a zapouzdřená identifikace
- archivace prostředí

► 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 verzi vše / kromě označeného / pouze označené

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

► 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)

► Integrační sestavení

- 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 označováno ve verzovacím systému

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

► Kusovník, archivace prostředí

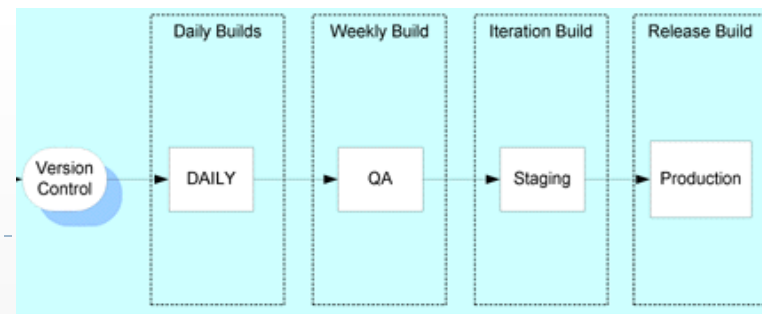
► Kusovník

- kompletní seznam prvků sestavení
 - reprodukovatelnost sestavení kdekoli, kdykoli
 - zejména při distribuovaném nebo jinak složitém buildu
- samoidentifikuji 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

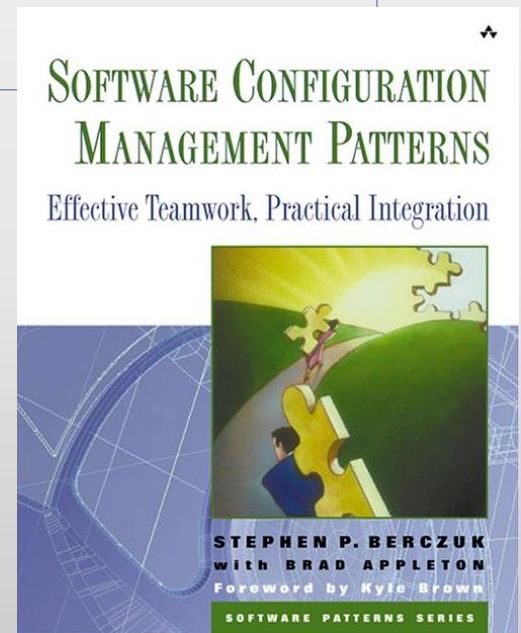


► Diskuse o sestavení

- Automatizace, autamatizace, 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

Rather than debugging, you can look at the changes since you last ran the [build].

Nejlepší praktiky: SCM+QA





☐ Způsoby zajištění kvality



► SCM + QA

► Typické workflow

- Daily Build and Smoke Test
- Continuous Integration

► Speciální workflow

- Release management





► **Daily Build** and Smoke test

► Integrovaní 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** „vždyt’ 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 disciplíny, 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 ;-)
- „Ix denně“ □ „neustále“
- Klíč: automatizace
 - co/ci, sestavení, testování, oznamování
 - robot na spouštění

cruisecontrol
continuous integration toolkit

Next Build Starts At:
June 21, 2002 15:30

05/07/2002 15:29:03 (1.1)
05/07/2002 04:25:35
05/07/2002 02:38:58
05/07/2002 01:03:55
04/25/2002 09:28:00 (2-2)

BUILD FAILED

Ant Error Message: E:\Projects\cvsv\cruisecontrol\main\sample_pro
messages should have been provided.

Date of build: 20020507023938

Time to build: 6 seconds

Last changed: 05/07/2002 04:25:33

Last log entry:

Errors/Warnings: (7)

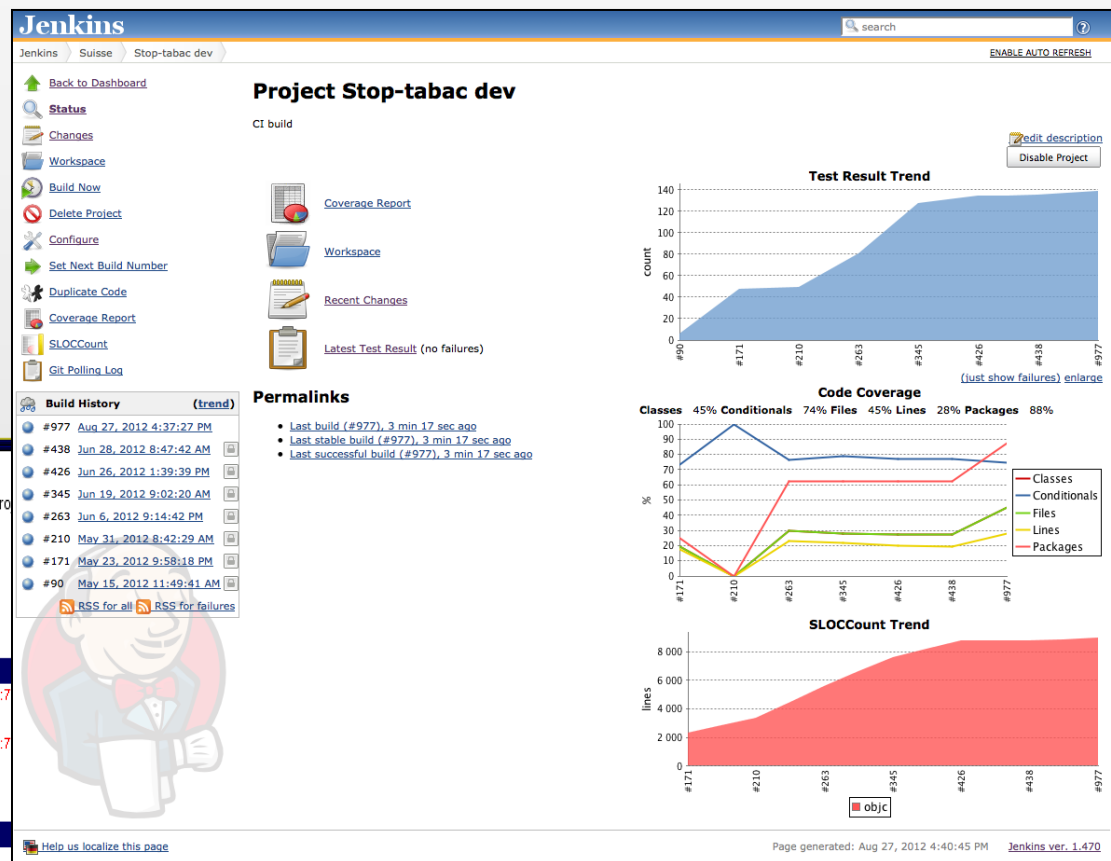
E:\Projects\cvsv\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java:7
?
^

E:\Projects\cvsv\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java:7
?
^

2 errors

Unit Tests: (1)

All Tests Passed



► Role v procesu sestavení

- Vývojář, tester / QA
- Deployment Engineer
- Operations



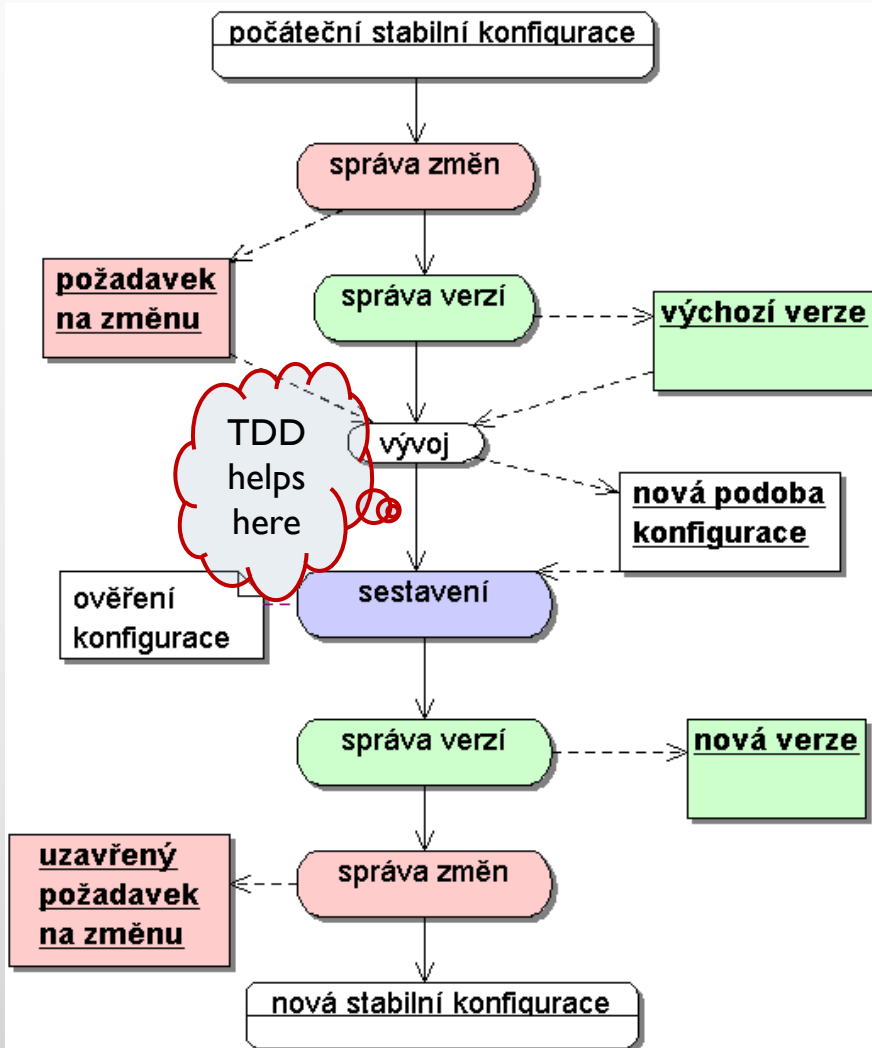
- DevOps





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 – krásna, jde se na oběd!

11:32 skleróza: bugzilla #11209 status: test