6. Techniky správy a organizace rozsáhlých softwarových projektů. Nástroje pro správu verzí a vývojových větví zdrojových kódů, nástroje pro automatické generování dokumentace a podporu orientace v rozsáhlých projektech. Infrastruktury zajišťující spolupráci mezi vývojáři navzájem a i s uživateli. Systémy pro sledování a řešení chyb a uživatelskou podporu.(A4M35OSP)

Motivace

Se správou a organizací softwarových projektů velmi úzce souvisí pojem ***konfigurační management (CM)***. Pod tímto pojmem se skrývá něco velmi praktického. Znamená to v každém okamžiku vědět, nebo být schopen identifikovat *verzi a stav* jakéhokoliv *artefaktu*. Artefaktem může být soubor se zdrojovým kódem, knihovna, pomocná utilita, dokument, diagram. Artefaktem je zkrátka soubor, který vznikl v průběhu vývoje. Kromě verze artefaktu musíme být schopni řídit změny, kterými artefakty prochází. Cílem CM je (Mrázek, 2011):

- verzování zdrojových kódů a dalších artefaktů
- mít přehled o dodaných verzích aplikace
- vědět, ve kterém prostředí je nasazená která verze aplikace
- vědět, v jakém stavu je které změnové řízení (požadavek na změnu)
- vědět, které změny kódu řeší které změnové řízení

Všechny agilní metodiky i zásady best practices předepisují v zájmu zajištění kvalitního softwarového procesu konfigurační řízení. Technicky je zajištěno zavedením *verzovacího systému* (SCM) a *issue tracking systému* (systémy Jira, Bugzilla) (Mrázek, 2011).

1 Nástroje pro správu verzí

1.1 Pojmy:

Centrální repozitář

Jedná se o centrální úložiště, ve kterém je umístěn projekt (na kterém může pracovat více vývojářů) a jeho historie. Distribuované systémy pro správu verzí nemají centrální repozitář. Veškeré informace jsou umístěny v každé pracovní kopii.

Working copy

Pracovní kopii projektu v požadované verzi je možné z centrálního úložiště získat pomocí operace checkout. Po úpravě lokální kopie je možné tyto změny odeslat do centrálního repozitáře pomocí operace commit.

Tag

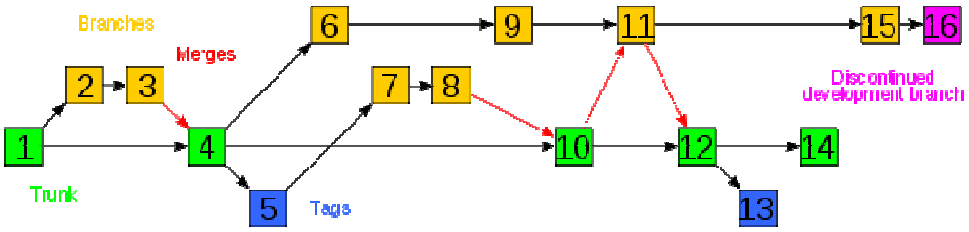
Pokud má některá revize zvláštní význam (např. se jedná o dodávku zákazníkovi), je možné tuto revizi pro přehlednost označit štítkem - tagem.

Branch & Merge

Když například po dodávce nějaké verze aplikace zákazník objeví chybu, je možné pro její řešení založit novou větev, přičemž v hlavní větvi (trunku) pokračuje dál vývoj nové verze aplikace. V momentě, kdy je chyba ve vedlejší větvi opravena, je tato větev s opravou sloučena (merge) s trunkem.

Dalším důvodem pro vytvoření vedlejší větve může být potřeba zásahu do aplikace bez rizika jejího rozbití.

Předcházející pojmy jsou znázorněny na Obrázek 1: DAG (Directed Acyclic Graph) s verzemi projektu.



Obrázek 1: DAG (Directed Acyclic Graph) s verzemi projektu

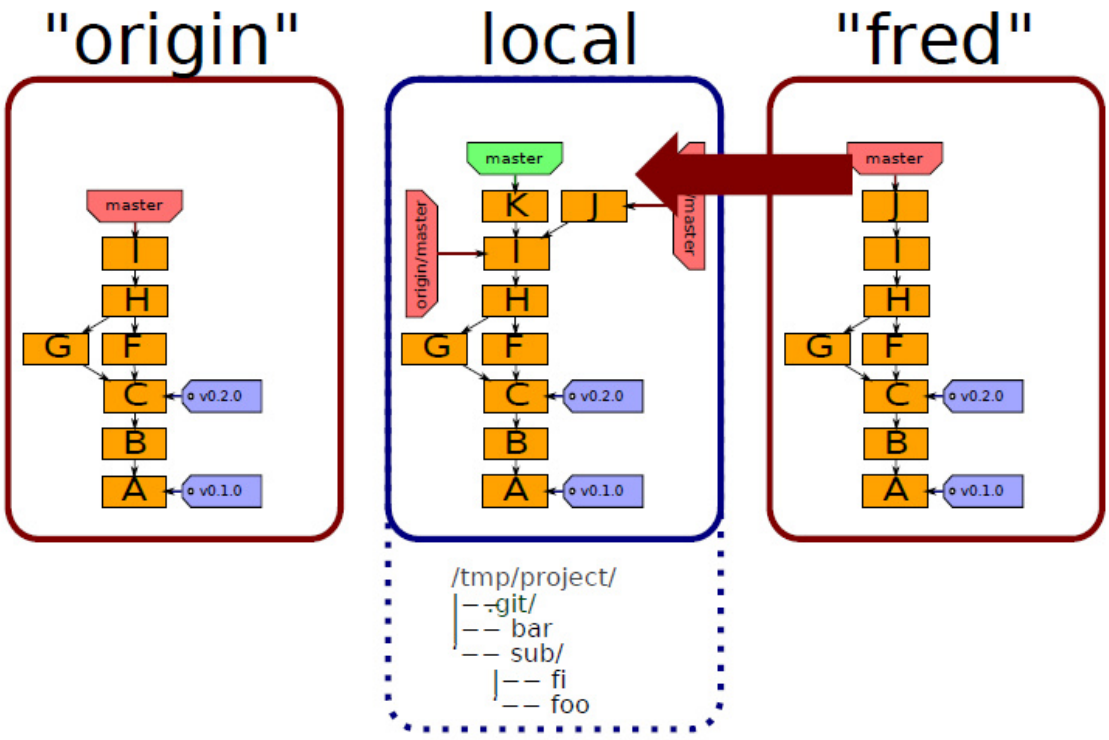
1.2 Motivace verzování zdrojového kódu

Verzování kódu umožňuje sdílení kódu mezi vývojáři, zálohování, paralelní vývoj v několika souběžných větvích, vrácení se ke konkrétní revizi kódu, zjištění autorství nebo zobrazení statistik. Bez verzovacího systému není možná efektivní spolupráce více vývojářů na jednom projektu. Verzovací systém přináší výhody i v případě, že na projektu pracuje jediný vývojář.

1.3 Vybrané verzovací systémy pro správu velkých projektů

GIT (Wikipedia, 2011)

GIT je distribuovaný nástroj pro správu verzí s důrazem na rychlost. GIT byl původně vyvinut Linusem Torvaldsem pro správu vývoje Linuxového kernelu. Každá pracovní kopie repozitáře je plnohodnotný repozitář s kompletní historií a plnohodnotnými možnostmi pro sledování revizí. Takový lokální repozitář nezávisí na připojení k síti či centrálnímu serveru. GIT je distribuován pod licencí GNU GPL v2.




```
$ git fetch fred
```

Obrázek 2: Peer-to-peer update

Dobrá podpora pro nelineární vývoj

GIT podporuje rychlé vytváření větví a slučování větví a obsahuje speciální nástroje pro vizualizaci a navigaci v nelineární historii projektu.

Distribuovaný vývoj

GIT dává vývojáři podobně jako verzovací nástroje Darcs, BitKeeper apod. lokální kopii kompletní historie vývoje a změny se kopírují z jednoho takového repozitáře do druhého. Tyto změny jsou importovány jako dodatečné vývojové větve a mohou být sloučeny stejným způsobem jako lokálně vyvinuté větve.

Kompatibilita s existujícími systémy/protokoly

Repozitáře mohou být zveřejněny pomocí http, FTP, rsync nebo pomocí protokolu GIT přes soket nebo ssh. GIT rovněž emuluje CVS server, což umožňuje použití existujících CVS klientů a pluginů do vývojových prostředí pro přístup do repozitářů GIT. Repozitáře Subversion a svk mohou být použity přímo pomocí git-svn.

Efektivní správa velkých projektů

GIT je velice rychlý a má dobrou škálovatelnost. Výkonové testy provedené společností Mozilla potvrdily několikanásobně vyšší rychlost GITu ve srovnání s jinými verzovacími nástroji. Získání

historie verzí z lokálního repozitáře GIT může být až stokrát rychlejší než jeho získání ze vzdáleného serveru. Důležité je, že nedochází ke zpomalení činnosti GITu s růstem historie projektu.

Kryptografická autentizace historie

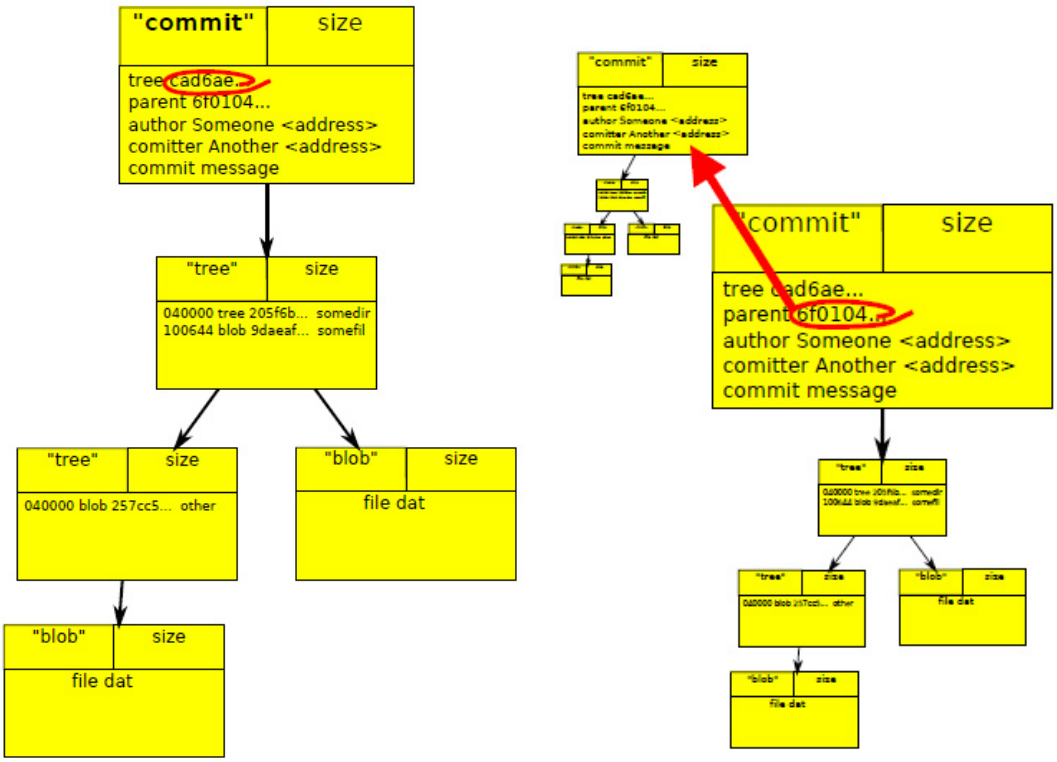
Historie v GITu je uložená takovým způsobem, že jméno konkrétní revize („commit“) závisí na kompletní historii vedoucí k této revizi. Po publikaci není možné změnit starou verzi, aniž by to zůstalo bez povšimnutí. Struktura je podobná hešovacímu stromu s dodatečnými informacemi v uzlech i listech.

KLADY

- Rychlost
- Menší nároky na diskový prostor
- Lepší podpora větvení
- Jednoduchý formát repozitáře, snadná oprava, k poškození dochází zřídka
- Snadná záloha

ZÁPORY

- Riziko vzdálení se jednotlivých repozitářů od sebe co se týče obsahu, dané autonomií distribuovaných repozitářů – nutnost řešit více konfliktů
- Náročnější na pochopení a ovládání pro začátečníky



Obrázek 3: Kryptografická autentizace historie

SVN - Apache Subversion

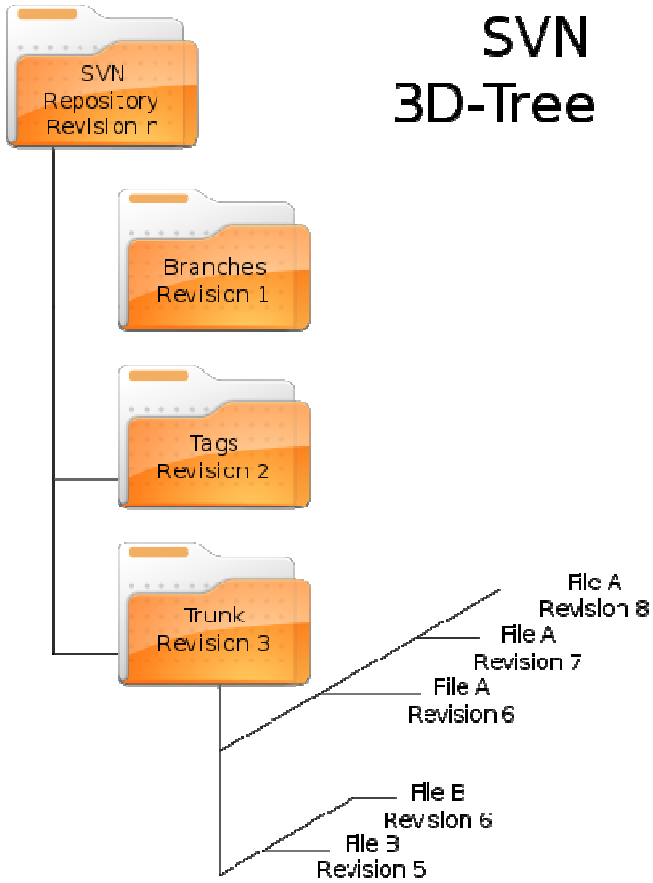
Jedná se o klasický a značně rozšířený systém pro správu verzí. Je založen na centrálním úložišti, ze kterého je možné získat pracovní kopii.

KLADY

- Pracovní verze jednotlivých vývojářů se nedostávají do tak odlišných stavů jako u distribuovaných systémů – vývojář motivován k častým aktualizacím své verze a odesílání svých změn do centrálního repozitáře
- Propracované uživatelské rozhraní včetně GUI nadstavby (např. Tortoise SVN)
- Snadné ovládání

ZÁPORY

- Single point of failure
- Nižší výkon a vyšší nároky na diskový prostor



Obrázek 4: Ilustrace stromu revizí v SVN (Wikipedia, 2011)

2 Nástroje pro automatické generování dokumentace

2.1 Javadoc

Javadoc je nástroj od společnosti Sun Microsystems pro generování dokumentace ze zdrojového kódu JAVA ve formátu HTML.

Formát „dokumentačních komentářů“ používaný v Javadoc je de facto průmyslovým standardem pro dokumentaci tříd v Javě. Některá IDE, jako např. Netbeans a Eclipse, automaticky generují Javadoc ve formátu HTML.

Javadoc taktéž poskytuje API pro vytváření docletů a tagletů, které umožňují analýzu struktury aplikace napsané v Javě. Takto funguje např. JDiff pro generování zpráv o změnách mezi dvěma verzemi nějakého API.

2.2 Doxygen

Doxygen je nástroj pro generování dokumentace pro programovací jazyky C, C++, C#, Fortran, Java, Objective-C, PHP, Python, IDL a další. Běží na většině unixových systémů (včetně Mac OS X) a Windows.

Doxygen je nástroj pro psaní referenční dokumentace pro software. Dokumentace je podobně jako u Javadoc napsána v kódu a je proto relativně snadné ji udržovat. Doxygen umí odkazovat mezi kódem a dokumentačními komentáři, takže čtenář může snadno dohledat konkrétní kus kódu.

Existuje velké množství dalších nástrojů pro generování dokumentace – viz http://en.wikipedia.org/wiki/Comparison_of_documentation_generators.

2.3 Enterprise Architect

Nástroj EA umožňuje generování různých typů UML diagramů (např. diagram tříd) přímo ze zdrojového kódu.

3 Infrastruktury zajišťující spolupráci mezi vývojáři navzájem a i s uživateli

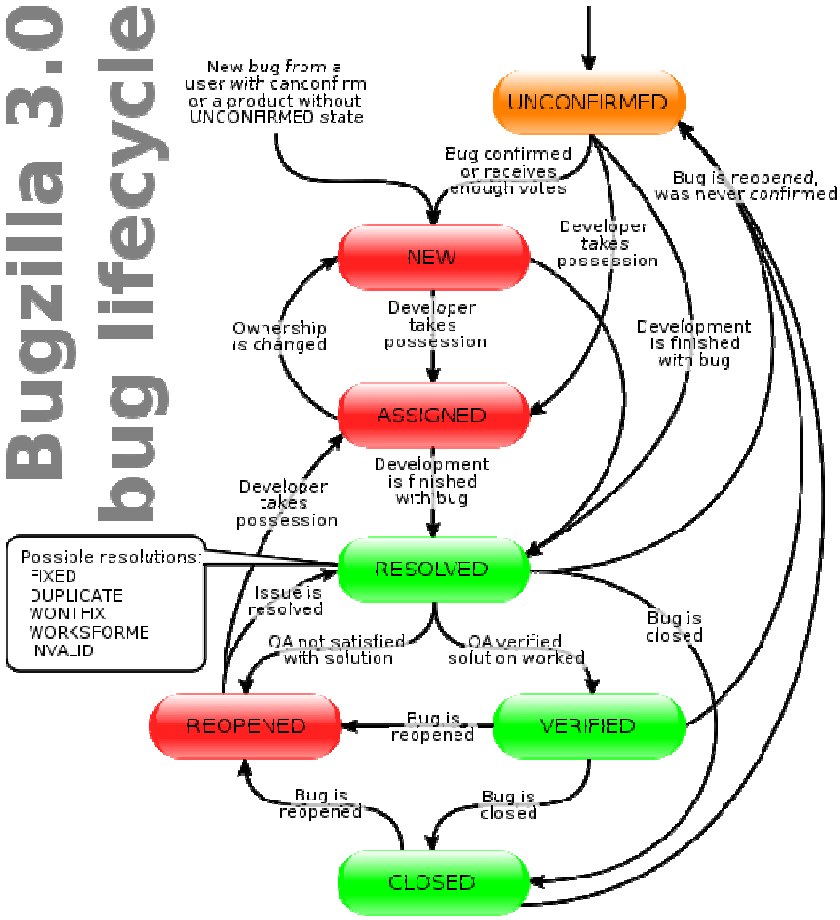
Projekt je možné hostavat na portále, který je k tomu určen. Jedná se například o javaforge.com. Takový portál nabízí následující služby pro celý tým pracující na projektu

- online SVN repozitář
- projekt management – úkoly, deadliny…
- sledování událostí a chyb
- kontinuální build
- wiki stránky
- přehledy a zprávy pro projektové manažery
- a mnoho dalšího

4 Systémy pro sledování a řešení chyb a uživatelskou podporu

4.1 Bugzilla

Bugzilla je webový víceúčelový nástroj pro sledování chyb(bugtracker), který byl původně vyvinut pro projekt Mozilla a zveřejněn pod Mozilla Public License. Dnes se jedná o open-source používaný velkým počtem organizací pro sledování chyb a událostí a příležitostně jako zdroj dalších informací pro projektový management. Obrázek 5: Životní cyklus chyby ukazuje životní cyklus chyby v Bugzille.



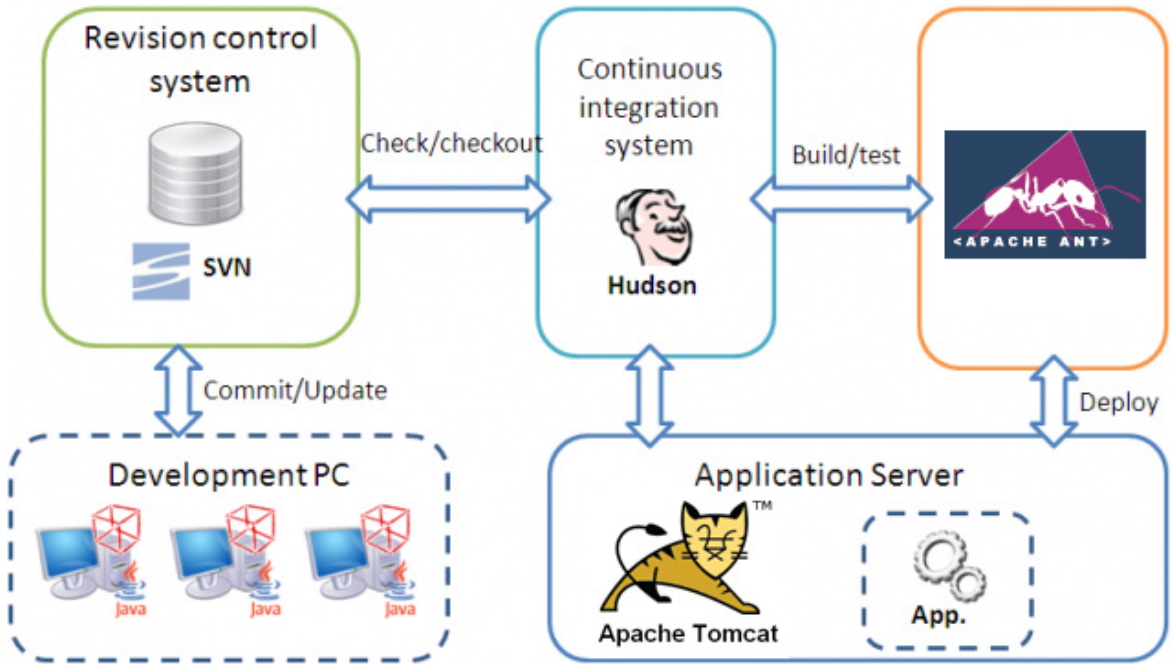
Obrázek 5: Životní cyklus chyby (Wikipedia, 2011)

4.2 Fóra, wiki stránky, helpdesk, hotline

Podle charakteru projektu je vhodné podporovat uživatele systému pomocí fór, wiki stránek nebo helpdesku. V případě placené podpory může být zřízena i hotlinka.

5 Integrační platforma (Mrázek, 2011)

Téměř vše výše uvedené spojuje platforma pro *kontinuální integraci* (CI). Ta umožňuje pravidelnou integraci zdrojových kódů všech vývojářů. Včasná identifikace rizik pomáhá předcházení problémům při dodání aplikace.



Obrázek 6: Kontinuální build pomocí Hudson (Mrázek, 2011)

Aplikační server na testovacím serveru hostuje systém *Hudson*. Ten lze nakonfigurovat pro integraci libovolného počtu projektů. Hudson si stáhne všechny zdrojové kódy z trunku *repository* (checkout). Pomocí *buildovacího nástroje* Apache Ant aplikaci *sestaví a nasadí na aplikační server* Tomcat nebo spustí sadu jednotkových nebo systémových testů. Výsledek buildu je přehledně dostupný uživateli. Spouštění projektů na integrační platformě lze naplánovat na libovolnou dobu. V případě neúspěšného buildu systém posílá email o chybě.

6 Bibliografie

Mrázek, Pavel. 2011. *Informační systém pro podporu výuky - Diplomová práce.* 2011.

Wikipedia. 2011. Apache Subversion - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia.* [Online] 2011. http://en.wikipedia.org/wiki/Apache_Subversion.

—. 2011. Git (software) - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia.* [Online] 2011. http://en.wikipedia.org/wiki/Git_%28software%29.

—. 2011. Wikipedia, the free encyclopedia. *Bugzilla - Wikipedia, the free encyclopedia.* [Online] 2011. http://en.wikipedia.org/wiki/Bugzilla.