Využitie podporných nástrojov pri vývoji softvéru v tíme

MARTIN NÁGL

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
mnagl@mtbiker.sk

Abstrakt. Dobre fungujúca kooperácia viacčlenného softvérového tímu nie je možná bez využitia nástrojov pre sledovanie úloh, odpracovaného času, konfigurácií, či verzií zdrojového kódu. Tieto nástroje poskytujú manažérovi tímu prehľad o stave projektu a informácie potrebné pre plánovanie prác. Vytvárajú tiež prostredie pre efektívnu spoluprácu vývojárov. V tejto eseji bližšie opíšem požiadavky na rôzne druhy podporných prostriedkov využívaných pri vývoji softvéru a ich použitie vysvetlím na príklade. Sprostredkujem skúsenosti s niektorými riešeniami získané z viacerých zdrojov a doplním vlastné skúsenosti s používaním rôznych nástrojov pri práci na softvérových projektoch v komerčnej spoločnosti.

Úvod

Tvorba softvéru je špecifickým výrobným odvetvím. Softvérové systémy sú čoraz komplexnejšie a ich vývoj si vyžaduje koordináciu početného tímu a viacerých špecializovaných činností. Členovia tímu medzi sebou potrebujú komunikovať a udržiavať si prehľad o postupe projektu. Je nepochybné, že monitorovanie a riadenie softvérového projektu sa nezaobíde bez využitia vhodných nástrojov.

V tomto príspevku sa pokúsim opísať oblasti, ktoré pokrývajú rôzne kategórie podporných nástrojov a vysvetliť ich význam pri riadení vývoja softvéru.

Nástroje pre riadenie verzií

Prvou skupinou nástrojov, ktoré sa začali využívať pri tvorbe softvéru už začiatkom 70. rokov 20. storočia, boli nástroje na riadenie verzií zdrojového kódu (*version control*). Historicky prvým systémom na riadenie verzií zdrojového kódu bol SCCS (Source Code Control System), vyvinutý v prostredí UNIXu v roku 1972. Až v 90.

Manažment v softvérovom inžinierstve, október 2006, s. 1-8.

rokoch bolo riadenie verzií integrované do vývojových prostredí a práca s ním sa tak podstatne zjednodušila.

Nástroje na riadenie verzií vo všeobecnosti umožňujú:

- jednoducho pridávať súbory a adresáre do databázy (*repository*), kde sa budú sledovať ich verzie a revízie
- skopírovať si pracovnú verziu jedného alebo viacerých súborov, vykonať v nich zmeny a následne vrátiť nové verzie do databázy (*check out / check in / commit*)
- získať aktuálnu, ale aj ľubovoľnú staršiu verziu súboru z databázy
- zvládnuť situáciu, keď s viacero používateľov súbežne vykonáva zmeny v rovnakom súbore
- spoločne označiť skupinu súborov a adresárov v aktuálnej verzii ako základ pre ďalšie zmeny, uchovávať históriu týchto označení
- získať z databázy kompletnú štruktúru adresárov a súborov v určitej verzii

Medzi najpoužívanejšie nástroje pre riadenie verzií zdrojových kódov patria v súčasnosti Subversion, CVS a Microsoft SourceSafe, existuje však aj mnoho ďalších produktov (spomeniem napr. Perforce).

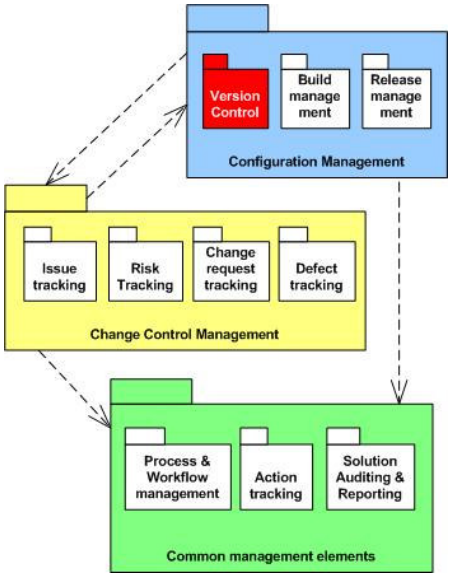
Martin Fowler píše v článku *Continuous Integration* [3] o systéme pre riadenie verzií ako o základnom kameni metodiky vývoja softvéru, kedy sa neustále udržiava aktuálna verzia zdrojového kódu vo funkčnom stave (bez kompilačných chýb). Každý vývojár vykonáva zmeny iba vo vlastnej (pracovnej) kópii zdrojových súborov a do centrálnej databázy (repository) uvoľní zmeny až vtedy, keď sú bezchybne integrované s existujúcimi zdrojovými kódmi. Tento princíp je dnes všeobecne používaný a pri vývoji v tíme je takmer samozrejmosťou.

Nástroje pre riadenie konfigurácií a zmien

Z riadenia verzií sa postupom času vyvinula rozsiahlejšia konceptuálna časť - riadenie konfigurácií. Samotné riadenie verzií by bolo možné realizovať pomerne jednoduchou aplikáciou. Systém pre riadenie konfigurácií je však komplexný nástroj, ktorý sa sústredí na vyvíjaný produkt ako celok a spravuje vytváranie jednotlivých vývojových verzií aplikácie (v jazyku programátorov sa často označujú ako *buildy*), a tiež finálnych verzií, ktoré sú určené pre koncových používateľov (*release*). Často ide o automatizované buildovacie systémy, ktoré sú schopné okrem zostavenia aktuálnej verzie aplikácie aj nasadiť ju do testovacieho prostredia a spustiť príslušné testy a aspoň čiastočne tak overiť jej funkcionalitu. Medzi systémy špecializované na riadenie konfigurácií patria napr. Ant, Maven alebo CruiseControl, v praxi som sa tiež stretol so zaujímavým nástrojom LuntBuild.

Doteraz som opisoval nástroje zamerané na sledovanie konkrétnych, nízkoúrovňových aspektov softvérového systému – zdrojových kódov, súborov, verzií, a pod. Pre sledovanie systému z opačnej perspektívy slúžia nástroje na riadenie zmien. Pri vytváraní systému je potrebné systematicky evidovať zoznam úloh (funkcionality), ktorá má byť implementovaná a tiež požiadavky zákazníka (zadávateľa). Počas práce na systéme je potrebné prideľovať zodpovednosť za riešenie každej čiastkovej úlohy konkrétnej osobe a sledovať životný cyklus každej čiastkovej úlohy. Pre potreby testovania je potrebné evidovať, ktoré funkcie sú zapracované v konkrétnych vývojových verziách a nakoniec, pri vydávaní finálnej verzie musíme mať prehľad o tom, či obsahuje všetky požadované funkcie.

Najvýraznejší prínos v procese vývoja softvérového produktu majú nástroje, ktoré integrujú riadenie zmien a riadenie konfigurácií. Dokážu teda priamo zaznamenať vzťah medzi úlohami (požiadavkami zákazníka), verziami vyvíjanej aplikácie a verziami zdrojového kódu.



Obr. 1. Vzťah medzi riadením konfigurácií a riadením zmien.

Príklad použitia nástroja na riadenie konfigurácií a zmien

Vysvetlime si úlohu integrovaného nástroja na riadenie konfigurácií a zmien na príklade. Predstavme si spoločnosť, ktorá vyvíja svoj softvérový produkt – napríklad grafický editor HyperSuperDraw. Spoločnosť pravidelne vydáva nové verzie editora. Vývojový tím pozostáva z analytika a návrhára, ktorý rozhoduje, ktoré funkcie budú implementované v nasledujúcej verzii, ďalej z niekoľkých programátorov, ktorí

implementujú priradené úlohy a z testovača, ktorý overuje správnosť funkcionality. Aktuálna verzia editora je HyperSuperDraw 2.1.

Používatelia editora HyperSuperDraw 2.1 sú s produktom veľmi spokojní, chýba im však jedna funkcia - export obrázku do formátu PNG. Analytik našej spoločnosti teda rozhodol, že vo verzii HyperSuperDraw 2.2 pribudne nová funkcia. Spoločnosť sa sústredí na kvalitu svojich produktov a používa Integrovaný nástroj na riadenie konfigurácii a zmien. Analytik si teda otvorí Integrovaný nástroj na riadenie konfigurácii a zmien (nazvime ho skrátene napríklad iNARKOZ) a zaznamená v ňom úlohu, ktorú je potrebné implementovať. S úlohou zaznamená aj popis zmeny, ktorá sa požaduje a doplní samozrejme funkčnú špecifikáciu. Následne pridelí úlohu na implementáciu niektorému z vývojárov. Systém iNARKOZ odošle vývojárovi notifikačný mail (napríklad s textom "Bola vám pridelená úloha HSD-345 na implementáciu"). Vývojár si prečíta popis úlohy v systéme iNARKOZ a začne pracovať. Na svojej pracovnej stanici si otvorí vývojové prostredie, načíta si aktuálnu verziu zdrojových súborov a pridá zopár potrebných riadkov kódu, napr. do súborov ImageExport.cpp a Menu.cpp. Následne zostaví aplikáciu a odskúša si svoju novú funkciu. Ak funguje a je s ňou spokojný, uvoľní upravené zdrojové kódy do spoločnej databázy (repository) – teda vykoná "check-in". Systém iNARKOZ je natoľko sofistikovaný, že pri "check-ine" zmien umožní programátorovi zadať informáciu, že tieto zmeny patria k úlohe HSD-345. Následne vývojár pomocou systému iNARKOZ automaticky vytvorí novú vývojovú verziu programu - HyperSuperDraw build 6543 – a zmení stav úlohy HSD-345 aby bolo zrejmé, že úloha čaká na otestovanie. Systém odošle testovačovi notifikačný mail s textom "Bola vám pridelená úloha HSD-345 na testovanie.". Testovač si prečíta popis úlohy v systéme iNARKOZ a začne pracovať. Spustí vývojovú verziu programu HyperSuperDraw build 6543 a podľa popisu úlohy a funkčnej špecifikácie otestuje úlohu. Ak nájde chybu, zmení stav úlohy v systéme a vráti ju programátorovi na prepracovanie. Ak je funkcionalita v poriadku, označí úlohu ako otestovanú.

Rovnakým procesom mohlo prejsť viacero úloh, ktoré sú v aktuálnej vývojovej verzii zapracované, otestované a čakajú na vydanie. V ideálnom prípade je v systéme iNARKOZ pri každej úlohe evidovaný aj čas, ktorý na nej odpracovali jednotliví riešitelia, čo umožňuje vedúcemu projektu sledovať čerpanie zdrojov. Pri vydaní finálnej (*release*) verzie sa aktuálna vývojová verzia označí ako HyperSuperDraw 2.2 a po schválení a prípadnom ďalšom otestovaní je pripravená na vydanie.

Na tomto jednoduchom príklade som sa pokúsil opísať základné prípady použitia nástroja na riadenie konfigurácií a zmien. Je možné využiť ho pre centralizované riadenie prác na projekte, evidenciu úloh, verzií produktu a evidenciu odpracovaného času. V praxi závisí životný cyklus úlohy a spôsob práce od konkrétneho projektu a metodiky v spoločnosti.

Nástroj by mal formalizovať vzájomnú komunikáciu členov tímu a sústrediť ju na jedno miesto. Je samozrejmé, že počas práce na konkrétnej úlohe je často potrebná priama komunikácia členov tímu. Napriek tomu je veľmi vhodné udržiavať miesto, kam sa budú aspoň stručne zaznamenávať poznámky k implementácii alebo testovaniu.

Ďalšie aspekty použitia nástrojov pre riadenie konfigurácii a zmien

Široká škála funkcií podporných nástrojov, ktoré sa v praxi využívajú, v konečnom dôsledku pomáha obmedziť riziká, ktoré by inak ohrozovali softvérový projekt z dôvodu zlého riadenia práce, neorganizovanej komunikácie v tíme či nedostatočnej informovanosti zodpovedných osôb o postupe projektu. Väčšina dostupných nástrojov poskytuje prehľady úloh podľa stavu riešenia, podľa riešiteľa, podľa verzie v ktorej majú byť zaradené, prehľady odpracovaných hodín podľa úloh a riešiteľov – sú to informácie, ktoré manažér projektu potrebuje pre správne rozhodnutia a vedenie projektu. Ťažko si dnes možno predstaviť tvorbu rozsiahleho informačného systému bez podobného podporného nástroja, nehovoriac o práci podľa medzinárodných štandardov (napr. ISO 9001) a získaní príslušného certifikátu kvality.

Je potrebné rozhodnúť, či sa bude nástroj na sledovanie zmien používať interne (iba používatelia z prostredia spoločnosti), alebo sa umožní prístup k nemu z externého prostredia (pre zákazníkov a používateľov produktu). V oboch prípadoch je potrebné evidovať chyby, ktoré sa v produkte odhalia až počas prevádzky. Je teda logické, ak sa spoločnosť rozhodne poskytovať podporu svojich produktov a dať používateľom možnosť zadať chybu priamo do svojej evidencie, typicky pomocou webového rozhrania. Integrovaný nástroj pre riadenie konfigurácií a zmien tak slúži ako tzv. *bug-tracker* a každá chyba (*bug*) sa rieši rovnako ako nová úloha, ktorá sa zadáva na implementáciu. Táto prax je bežná pri open-source, ale aj komerčných projektoch.

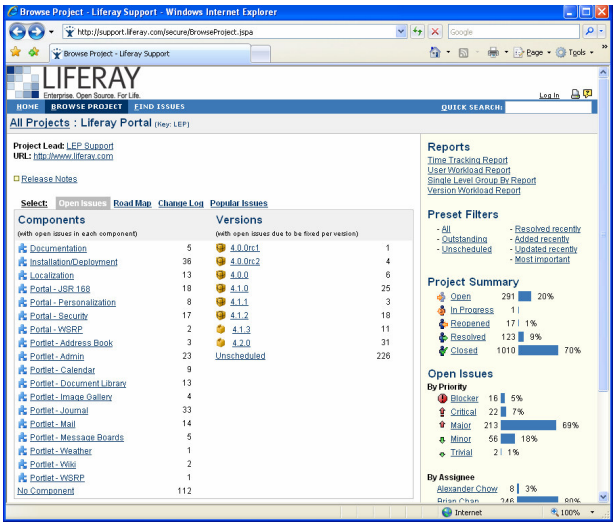
Skúsenosti s konkrétnymi riešeniami

Na trhu existuje veľké množstvo voľne dostupných aj komerčných produktov integrujúcich manažment projektov, verzií, konfigurácií a úloh. Zo systémov, ktoré som mal možnosť vidieť, hodnotím najvyššie produkt JIRA od spoločnosti Atlassian, ktorý však pre komerčné aj akademické účely vyžaduje platenú licenciu. Pre open-source a neziskové projekty je možné získať licenciu zdarma. Tento systém som používal pri práci s portálovým systémom Liferay Portal v rámci záverečného projektu bakalárskeho štúdia na vyhľadávanie známych problémov a hlásenie objavených chýb.

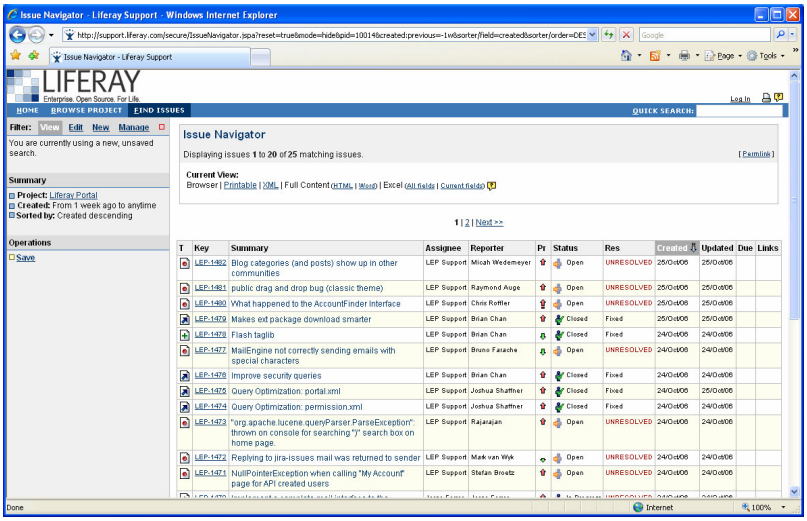
V rámci tímového projektu používame systém Dotproject. Tento produkt má rozsiahle možnosti evidencie a sledovania úloh, odpracovaného času a postupu projektu, chýba mu však prepojenie s riadením verzií. Použiteľnosti Dotproject-u by prospeli drobné úpravy používateľského rozhrania.

V spoločnosti Softec, kde pracujem, používame vlastné nástroje pre konfiguračné riadenie a riadenie zmien. Tieto nástroje sú prispôsobené metodike, ktorá sa používa v Softecu a postupne sa vyvíjajú, aby ich bolo možné nasadiť v projektoch s novými požiadavkami. Aj iné softvérové spoločnosti môžu byť postavené pred otázku – použiť niektorý z dostupných produktov, alebo vyvinúť vlastný? Na túto otázku neexistuje jednoznačná odpoveď, treba však povedať, že funkcionalita existujúcich produktov je na veľmi dobrej úrovni a pri rozhodovaní treba preto začať prieskumom už hotových riešení.

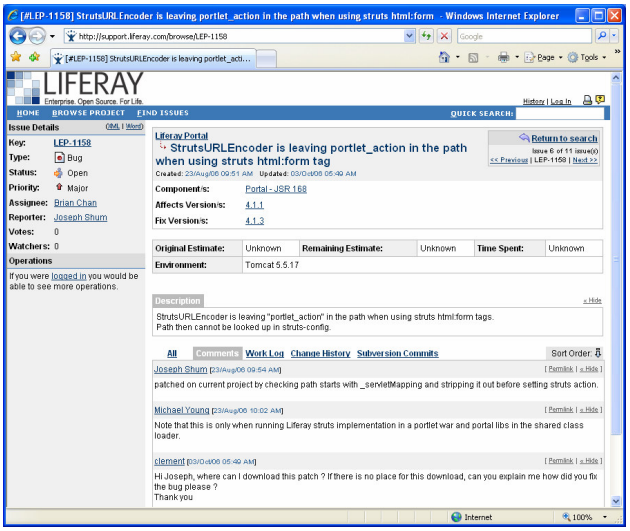
Na záver by som rád ukázal niekoľko obrazoviek zo systému na správu konfigurácii a zmien – JIRA.



Obr. 2. JIRA - Prehľad projektu Liferay Portal



Obr. 3. JIRA – Zoznam úloh a chýb v projekte Liferay Portal



Obr. 4. JIRA – Detail jednej z úloh v projekte Liferay Portal

Záver

Podporné nástroje umožňujú riadenie rozsiahlych softvérových projektov. Bez organizovanej databázy úloh a chýb nie je možné dodať kvalitný produkt. Pre použitie konkrétneho nástroja je dôležité, aby vyhovoval potrebám konkrétneho projektu a aby nevyžadoval zbytočné zmeny v pracovných postupoch. Používatelia by mali jasne vidieť jeho prínos a nevnímať ho ako ťažkopádny systém, ktorý im prináša viac problémov ako úžitku. Stav, kedy je nástroj na sledovanie úloh dokonale využitý, spoznáme podľa toho, že každý z členov tímu má v prehliadači nastavenú ako domovskú stránku zoznam svojich nevyriešených úloh a jeho tajným snom je mať možnosť zadať správcovi kancelárie úlohu, aby pravidelne objednával zamestnancom pizzu.

Použitá literatúra

1. Briggs, L. L.: Software Configuration Management: New Tools to Streamline Development, *Application Development Trends*, 31.03.2005 http://www.adtmag.com/article.aspx?id=10945
2. Edwards, C.: Moving management to benefit from Configuration Management over Version Control. *Configuration Management Crossroads*, 30.06.2006 http://www.cmcrossroads.com/content/view/6780/135/

3. Fowler, M.: Continuous Integration, 01.05.2006 http://www.martinfowler.com/articles/continuousIntegration.html
4. Sayko, M.: Using Process-Enabled SCM Tools to Facilitate the Software Development Lifecycle. *Configuration Management Crossroads,* 17.05.2005 http://www.cmcrossroads.com/content/view/6691/135/
5. Spolsky, J.: Painless Bug Tracking, *Joel on Software,* 08.11.2000 http://www.joelonsoftware.com/articles/fog0000000029.html

Annotation

Tools to support software development in team

Tools for tracking issues, time worked, software configurations or source code versions are essential for good software team cooperation. These tools provide project state overview for project manager and necessary information for project planning. They also create an environment for effective cooperation of developers. In this essay, I describe requirements for various kinds of tools used in software development and explain their use on a real-world example. I mention experience with certain products obtained from various sources and add my own experiences from working in a commercial software company.