SPRÁVA VERZIÍ, SLEDOVANIE ÚLOH A DOKUMENTÁCIA. SPOLU ALEBO KAŽDÝ ZVLÁŠŤ?

Výber správnych vývojárskych nástrojov a ich integrácia je prvý krok k premene nápadu do kvalitného softvéru.

Michal Dorner

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
dorner.michal[zavináč]gmail[.]com

Abstrakt. *Systémy na správu verzií a sledovanie úloh patria medzi základné podporné prostriedky vývoja softvéru. Spolu s dokumentáciou bývajú súčasťou každého väčšieho projektu. Táto esej sa zaoberá úvahami nad ich vzájomnou integráciou, ergonómiu a správnym používaním. Uvádza aktuálnu situáciu a funkcionalitu, o ktorú by sa tieto systémy mohli vylepšiť pre lepšiu orientáciu v celom projekte.*

Kľúčové slová: *podporné prostriedky, správa verzií, sledovanie úloh, commit message, anotácie*

Úvod

Medzi základné prostriedky pre podporu vývoja softvéru patria systémy na správu verzií a systémy na sledovanie úloh. Neoddeliteľnou súčasťou projektov býva aj dokumentácia, často vo forme wiki stránok [2]. Programátori pritom stále obľubujú aj používanie jednoduchých anotácií priamo v zdrojovom kóde. Každý z týchto systémov sa dá používať samostatne, čo sa aj zvyčajne deje. Nebolo by však lepšie tieto systémy prepojiť? A aké je vlastne vhodné použitie jednotlivých systémov?

Sledovanie úloh, správa verzií a dokumentácia

V systéme na sledovanie úloh jednoducho dokumentujeme všetko čo treba urobiť, kto to má urobiť a kedy, respektíve v akom je to stave. Pod to spadá napríklad požadovaná

funkcionalita výsledného softvéru alebo požiadavky na odstránenie chýb. Jednotlivé úlohy sa potom zvyčajne rozdeľujú na menšie podúlohy, pri ktorých je jasnejšie čo konkrétne sa má spraviť. Potom vieme aj presnejšie odhadnúť koľko to bude trvať.

Správa verzií (angl. version control system - VCS) dokumentuje vývoj zdrojových kódov, pričom zaznamenáva jednotlivé zmeny a ich autora. S každou novou verziu sa zvyčajne zadáva aj krátky komentár (angl. commit message), približujúci, čo sa danou novou verziu rieši.

Dokumentácie máme technické a používateľské. V tomto dokumente sa budem ďalej zaoberať iba technickou. V nej by sa mali nachádzať smerodajné opisy štruktúry softvéru, použitých algoritmov, rozhraní a jednotlivých modulov alebo tried.

Aký je súvis medzi verziami, úlohami a dokumentáciou?

Osobne to vidím tak, že kód sa vyvíja na základe stanovených úloh a jednotlivé úlohy menia svoj stav na základe zmien v kóde. Zmena v zdrojových kódoch sa pritom môže prejaviť aj na zmene dokumentácie. Zjednodušene sa dá povedať, že závisieť môže všetko od všetkého. Samozrejme nemusí platiť, že každá úloha súvisí zo zdrojovým kódom softvéru.

Prepojenie systémov

Ak akceptujeme načrtnuté vzájomné prepojenia, otázkou ostáva ako ich realizovať. Myslím si, že ideálny systém by bol taký, v ktorom sa prostredníctvom niekoľkých klikov myši vieme voľne pohybovať medzi úlohou, súvisiacimi zmenami kódu v správe verzií a dokumentáciou, napríklad wiki stránkami.

Pre vývojárov by bol takýto systém nesmierne užitočný. Ušetril by mnoho času hľadaním potrebných informácií na troch rôznych miestach a tým uľahčil napríklad zorientovanie sa v práci iného člena tímu, alebo aj vlastnej práci, ak sa k nej vraciame po dlhšom čase. Takéto systémy pritom už z časti existujú.

Commit message

Prepojenia medzi úlohami a verziami sú už bežné a realizujú sa prostredníctvom špecificky formátovaných komentárov, ktoré sa zadávajú pri odosielaní novej verzie do správy verzií. Zvyčajne je potrebné uviesť identifikátor danej úlohy. Tento commit, sa potom zobrazuje v zozname pri prehliadaní danej úlohy. Niektoré systémy ešte umožňujú automatické menenie stavu danej úlohy ak komentár obsahuje určený text, napríklad *close* pre zatvorenie úlohy. Bol by som veľmi rád, ak by takéto prepojenia fungovali aj v prípade odkazov na wiki stránky a rovnako by boli užitočné aj prepojenia na iné verzie v rámci správy verzií. Problémom ostáva nie najlepšia ergonómia. Pamätať si identifikátory úloh sa nedá a nikomu sa nechce zakaždým otvárať správu úloh aby ho z tadiaľ skopíroval. Privítal by som vzájomnú integráciu správy verzií a sledovania úloh priamo v IDE, kde by sa kód úlohy dal do commit message vložiť cez jednoduché fulltextové vyhľadávanie.

Okrem toho by som si vedel predstaviť aj systém, v ktorom označím predchádzajúci commit, ktorého pridaný kód idem významne zmeniť. To už síce existuje aj v súčasnosti, no ja by som to ešte rozšíril o možnosť zaregistrovania odchytávania takýchto zmien.

Vývojárovi, ktorý by si takéto odchytávanie zaregistroval na príslušný commit, by automaticky prišlo upozornenie, ak sa daná časť kódu zmenila. Pochopiteľne sa nedá spoľahnúť, že dotyčný čo daný kód ide zmeniť, to vždy správne označí, ale spolu s integračnými testami by to mohlo prispieť k skorému odhaleniu množstva chýb.

Anotácie

Akoby už toho nebolo dosť, sú aj ďalšie možnosti ako zlepšiť prepojenie zdrojového kódu s úlohami. Myslím si, že programátori v niektorých prípadoch radšej používajú anotácie v zdrojovom kóde ako rôzne externé systémy. Pridanie anotácie je totiž výrazne jednoduchšie než vytvorenie novej úlohy v systéme na správu úloh. Platí to hlavne ak sa jedná o relatívne triviálne úlohy typu *TODO* alebo *FIXME*, ktoré sa dajú opísať niekoľkými slovami. Problémom anotácii však je, že nakoľko sú iba v zdrojovom kóde a nevidíme ich pri správe úloh, tak často ostávajú zabudnuté.

Témou anotácií v zdrojovom kóde a ich prepojení s úlohami sa zaoberali aj autori Anvik a Storey. Vo svojom článku [1] predstavili systém, ktorý umožňoval prepájanie neformálnych anotácií s formálnym zápisom v správe úloh. Aj keď nimi predstavený systém je značne obmedzený a nespolupracuje so správou verzií, inšpiroval ma k nasledujúcej úvahe.

Ak by sa po odoslaní zmien do centrálneho repozitára, ktoré pridávajú alebo odstraňujú anotácie, automaticky vytvárali a zatvárali úlohy, získali by sme výhody oboch prístupov. Pre vytvorenie jednoduchých úloh by ani nebolo potrebné opustiť textový editor. Navyše by obsahovali prepojenie na konkrétne miesto v zdrojovom kóde. Nové úlohy by sa štandardne vytvárali ako podúlohy k úlohe, na ktorú odkazoval commit message a priraďovali by sa danému vývojárovi, ktorý zmeny odoslal. Úloha by sa potom automaticky uzavrela ihneď po odstránení prislúchajúcej anotácie z kódu.

Takýto systém by bol pohodlný pre vývojárov a viedol by ich k zaznamenávaniu aj malých podúloh. Pritom by stále zabezpečoval, že sa daná úloha nestratí a len tak sa na ňu nezabudne. Ak by sa po automatickom vytvorení úlohy ukázalo, že úloha nie je taká triviálna ako sa pôvodne zdalo, v systéme na správu úloh by sa úloha dala samozrejme jednoducho odpútať od anotácie a upraviť.

Waypoints

V článku [1] sa vyskytol aj pojem *waypoints*, v zmysle iného názvu pre anotácie. Tento názov má zdôrazňovať, že sa jedná o jedinečne určené body v zdrojovom kóde, ktoré môžu slúžiť na navigačné účely. Toto by veľmi dobre zapadalo do celkového konceptu prepojení medzi zdrojovým kódom, úlohami a dokumentáciu. V neposlednom rade by takéto odkazy veľmi dobre fungovali aj v rámci krížového odkazovania v samotných komentároch v zdrojovom kóde.

Použitie by bolo veľmi jednoduché, stanovil by sa špecifický formát, ktorý by určoval že daný komentár je *waipoint* a čo z neho je jeho identifikátor. Potom by sa tento identifikátor dal používať kdekoľvek v dokumentácii, opise úloh, commit message alebo aj v iných komentároch v zdrojovom kóde, pričom by vždy odkazoval na dané miesto, kde sa v aktuálnej (alebo zvolenej) verzii nachádza.

Ergonómia používania

Pri všetkých spomenutých pravidlách práce so správou verzií a systémom na správu úloh a ich vzájomným prepájaním, sa už môže vyskytnúť otázka, či toho už nie je priveľa. Všetky tieto nástroje by mali uľahčovať monitorovanie vývoja projektu a pritom súčasne by mali byť užitočné aj samotným programátorom. Akonáhle bude systém príliš komplikovaný a nepohodlný na používanie, nikto ho nebude používať. To ale neznamená, že treba okresať funkcionalitu. Treba ju len vhodne zakomponovať do vývojového prostredia.

Odkazovanie na úlohy cez commit message sa už bežne používa aj teraz, je však potrebná istá disciplína programátorov. Zakaždým si treba pozrieť identifikátor úlohy, čo je zvyčajne „ľahko zapamätateľné a veľavravné“ číslo. Realita je taká, že často sa namiesto odkazovania na úlohu odošle komentár typu „opravilo sa niekoľko chýb“. Jednoduchým riešením by podľa môjho názoru bolo zakomponovanie správy úloh aj verzií priamo do vývojového prostredia. To by si cez vytvorené služby získalo zoznam všetkých úloh (prípadne len priradených ku prihlásenému používateľovi) a pri zadávaní commit message by umožnilo medzi nimi rýchle fulltextové vyhľadávanie. Rovnaký systém by pritom mohol fungovať aj pri odkazovaní na wiki stránky.

Používanie anotácií na vytváranie menších podúloh je veľmi jednoduché a oproti tomu na čo sú programátori zvyknutý už teraz, nevyžaduje vôbec žiadnu prácu navyše. Vzhľadom na túto jednoduchosť a pritom vysokú pridanú hodnotu, akú by toto prepojenie anotácií do systému na správu úloh prinieslo, myslím si že toto je jedna z prvých vecí, ktorá by sa naozaj mohla začať používať.

Náročnejšie na implementáciu aj používanie by bolo odkazovanie na predchádzajúci commit, ktorého kód sa programátor chystá zmeniť. Problémov je tam hneď niekoľko. Nie vždy sa dá jednoznačne určiť ktorý predchádzajúci commit idem zmeniť a hlavne vyžaduje to prácu navyše. Programátor by musel manuálne vyhľadať identifikátor pre daný commit, nakoľko v tomto prípade nijaké fulltextové vyhľadávanie nepomôže. Takýto systém by sa teda mal používať na čisto dobrovoľnej báze a len v prípadoch keď to má zmysel a uchovanie informácie má väčšiu hodnotu ako navyše strávený čas.

Čo do správy verzií nepatrí?

Štandardne by Commit message mala obsahovať hlavne informáciu čo daný commit pridal, opravil alebo zmenil a v akom je to štádiu (napr. hotové alebo rozpracované). Ja by som ku tomu ešte pridal prepojenia na súvisiace úlohy a dokumentáciu. Určite však tieto správy nie sú vhodné na uchovávanie komplexnejších informácií, napríklad opisov algoritmov, použitia, alebo závislostí na iných častiach systému. Pri prehliadaní repozitára, totiž štandardne vidíme len niekoľko posledných commitov. K tým predchádzajúcim sa síce vieme ľahko dostať, no nevidíme ich na „prvý pohľad“ a hlavne nie sú nijako štruktúrované. Takéto informácie je určite lepšie napísať do dokumentácie (napr. wiki) a do samotného komentáru pridať iba odkaz na ne. Niektoré veci je pritom stále najlepšie písať (aj) do komentárov priamo v zdrojovom kóde, kde sú ihneď na očiach každému, kto ide s daným kódom pracovať.

Komentáre pritom nie sú jediná vec zo správy verzií, ktorá býva nesprávne používaná. Hlavne začiatočníci majú v obľube mať pod správou verzií celý projektový adresár zo všetkými súbormi. Ten zvyčajne obsahuje aj skompilované zdrojové kódy, rôzne logy zo zotavovacieho procesu alebo dočasné súbory z IDE. To všetko pod správu verzií určite nepatrí. Negatívne sa to prejaví najmä pri distribuovaných VCS, v ktorých sa uchováva na každom počítači kompletná história verzií. Distribuované VCS sú pritom čím ďalej populárnejšie a používanejšie [3], takže na to treba myslieť. Ak chcete archivovať aj výsledné „binárky" (výsledné preložené a spustiteľné programy aj s dátami), použite na to iný nástroj, napríklad ich dajte na obyčajný súborový server a správne ich pomenujte.

Záver

Základnými podpornými prostriedkami pre vývoj softvéru sú systémy na správu verzií a sledovanie úloh. Veľmi užitočným doplnkom býva aj dokumentácia vo forme wiki stránok. Vzájomná integrácia týchto systémov je rádovo zložitejšia ako ich samostatné používanie, prináša však oveľa lepšiu orientáciu v celom projekte. Vzájomnú integráciu však musia podporovať tak samotné systémy, ako aj jednotliví vývojári. Ak nebudú do systému vkladať potrebné informácie, systém ich nemá z kade získať.

Základom pre vzájomnú integráciu systémov je uvádzanie odkazov na úlohu alebo dokumentáciu v rámci commit message. V systéme na správu úloh potom vieme mapovať prácu na úlohe k jednotlivým zmenám v zdrojovom kóde.

Vzhľadom na to, že pre vytvorenie novej úlohy treba opustiť textový editor, zapnúť externý systém a v ňom úlohu v niekoľkých krokoch „vyklikať", programátori často menšie podúlohy do systému nezaznamenávajú vôbec. Riešením tohto problému je nechať ich používať anotácie, na ktoré sú zvyknutí. Systém by mal pritom tieto pridané alebo odobraté anotácie rozoznávať a podľa nich automaticky vytvárať a zatvárať prislúchajúce úlohy.

Anotácie by ďalej mohli fungovať aj ako identifikátory miesta v zdrojovom kóde. Na tieto miesta by bolo možné sa odkazovať kdekoľvek v dokumentácii, opise úloh, commit message alebo aj v iných komentároch v zdrojovom kóde.

Aj pri úvahách o komplexných systémoch na správu projektu, netreba zabúdať na správne použitie jeho jednotlivých častí, najmä správy verzií. Commit message by nemal obsahovať priveľa informácií (skôr odkazy na ne), skompilované a iné generované súbory väčšinou pod správu verzií nepatria a nepatrí sa odosielať do hlavnej vetvy nefunkčné verzie.

Použitá literatúra

1. Anvik, J., Storey, M.A.: Task articulation in software maintenance: Integrating source code annotations with an issue tracking system. *Software Maintenance, 2008. ICSM 2008. IEEE International Conference*, 460-461
2. Brown, M.K., Huettner, B., James-Tanny, C.: Choosing the Right Tools for Your Virtual Team: Evaluating Wikis, Blogs, and Other Collaborative Tools. *Professional Communication Conference, 2007. IPCC 2007. IEEE International*, 1-4

3. de Alwis, B., Sillito, J.: Why are software projects moving from centralized to decentralized version control systems? *Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Workshop*, 36-39
4. Callahan, J.R., Khatsuriya, R.R., Hefner, R.: Web-based issue tracking for large software projects. *Internet Computing, IEEE*, Vol. 2, No. 5 (1998), 25-33
5. Hinsen, K., Läufer, K., Thiruvathukal, G.K.: Essential Tools: Version Control Systems. *Computing in Science & Engineering*, Vol. 11, No. 6 (2009), 84-91
6. Johnson, J.N., Dubois, P.F.: Issue tracking. *Computing in Science & Engineering*, Vol. 5, No. 6 (2003), 71- 77

Annotation

Version control, issue tracking and documentation. Together or separately?

Version control and issue tracking system are now both fundamental support tools for software development. Along with documentation, they are part of any larger project. This essay focus on their integration, ergonomics and proper use. Lists the current situation and the functionality which can improve it for better orientation in the entire project.