OD ANOTÁCIÍ ÚLOH V KOMENTÁROCH K RIADENIU PROJEKTU: SLEDOVANIE ÚLOH

Vzájomné prepojenie týchto dvoch oblastí má potenciál sa v budúcnosti rozvíjať.

Martin Labaj

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
martin[.]labaj[zavináč]computer[.]org

Abstrakt. *Sledovanie úloh je v rámci softvérových projektov nevyhnutnosťou a preto pre tento účel existujú mnohé podporné prostriedky. Existujú však situácie, kedy sa členovia tímu vyvíjajúceho projekt, najmä programátori píšuci kód, neobrátia na externé nástroje, ale z rôznych dôvodov vytvárajú poznámky ohľadom úloh priamo v zdrojovom kóde vo forme komentárov. Prirodzeným spôsobom tak vzniká samostatný prostriedok na sledovanie úloh – anotácie úloh v zdrojovom kóde. Okolo tohto prostriedku následne vznikli nástroje na jeho podporu. V tejto eseji sa zamýšľam nad okolnosťami, prečo a v akých situáciách vývojári uprednostňujú anotácie pred dostupnými externými nástrojmi, ktoré sú už dokonca v projekte použité. Na základe týchto úvah ďalej zvažujem úlohu anotácií pri práci v tíme a teda aj úlohu, ktorú tento prostriedok zohráva alebo môže zohrať v rámci manažmentu projektu. Pri uvažovaní softvérových projektov je táto esej zameraná na malé tímy aplikujúce agilný vývoj softvéru metódou Scrum.*

Kľúčové slová: *komentáre, anotácie, podporné prostriedky, sledovanie úloh, agilný vývoj, Scrum*

Komentáre v zdrojovom kóde a riadenie projektu – existuje tu súvislosť?

Komentáre v zdrojovom kóde softvéru sú bežnou súčasťou softvérových projektov. Tieto komentáre sú často vnímané ako vysvetlivky alebo dokumentácia, no taktiež si programátori prostredníctvom nich zaznamenávajú rozličné úlohy. Vzniká tak podporný prostriedok – anotácie úloh priamo v zdrojovom kóde, ktoré majú potenciál aj z pohľadu riadenia projektov v softvérovom inžinierstve. Vzájomné prepojenie týchto dvoch oblastí nachádzajúcich sa v softvérovom projekte na veľmi odlišných úrovniach je zaujímavá téma, ktorá má v súvislosti s čoraz väčšou integráciou rôznych podporných prostriedkov a vzostupom popularity agilných metód vývoja softvéru potenciál v budúcnosti sa rozvíjať.

Prečo agilné metódy? Vývoj softvéru pomocou aplikácie agilných prístupov má svoje určité špecifiká a z dôvodov, ako je napríklad krátky produkčný cyklus [1], je v ňom podľa mňa potrebné klásť väčší dôraz na úlohy. Jedným z prístupov agilného vývoja je metóda Scrum, ktorej významnou súčasťou je riadenie úloh a je dôležité pri aplikácii tohto prístupu úlohy starostlivo sledovať. Prepojenie nástrojov pre podporu agilných metód alebo metódy Scrum s anotáciami úloh v zdrojovom kóde by teda mohlo podľa môjho názoru priniesť pozitívny efekt pre riadenie projektu. Preto má zmysel sa nad prostriedkom anotácií úloh a nástrojmi pre jeho podporu hlbšie zamyslieť a zvážiť jeho používanie.

Ako prostredie pre tieto úvahy som z uvedených dôvodov zvolil agilný vývoj. Venujem sa pri tom projektom vytváraným malými tímami. Menšie tímy sa podobajú školským tímom, môžem tak vziať do úvahy moje skúsenosti z predchádzajúcich projektov. Tiež sú v menšom tíme užšie vzťahy medzi jednotlivými členmi, čo má vplyv aj na obsah vytváraných komentárov v zdrojovom kóde.

Od komentárov k anotáciám úloh

Takmer každý programátor sa s komentármi zdrojového kódu stretol už vo svojich začiatkoch hneď pri získavaní prvých znalostí v programovaní. Podľa mojich skúseností, nielen z viacerých absolvovaných kurzov programovania, ale aj z rôznych publikácií, sú začínajúci programátori v týchto prípadoch vedení k chápaniu a používaniu tohto prostriedku ako možnosti pre vysvetlenie algoritmu zachyteného v zdrojovom kóde a teda aj svojich myšlienok. Takéto „tradičné“ použitie komentárov primárne slúži na zjednodušenie porozumenia programu, čo v prípade začiatočníkov znamená najmä použitie pre vlastné potreby, čiže neskoršie pochopenie, čo to vlastne vytvorili. Len málo rozsiahlejších softvérových projektov však vytvára jednotlivec, individuálny programátor. Softvérový projekt často vzniká v tíme, kde sa dostáva ku slovu kolaborácia. V takom prípade umožňujú komentáre použité v tomto základnom ponímaní členom tímu ľahšie pochopiť danú súčasť projektu, čím zjednodušujú spoluprácu. A aj v prípade, že na projekte pracuje jediný človek, je možné, že zdrojové kódy bude prezerať niekto ďalší a to v prípade, že projekt prevezme tím alebo jeho zdrojové kódy budú dokonca zverejnené. Kým dokumentácia projektu opíše projekt na vyšších úrovniach, priamo vysvetlenie jednotlivých častí kódu dokážu najlepšie poskytnúť komentáre a tak aj pri tejto ich dokumentačnej funkcii zohrávajú dôležitú úlohu pre zdieľanie kódu v tíme.

Použitie komentárov však samozrejme nie je obmedzené na konkrétne účely a je na programátorovi, čo napíše medzi znaky ohraničujúce komentár. Preto je prirodzené, že programátori tento prostriedok používajú nielen na vysvetľovanie zdrojových kódov, ale zároveň si pomocou neho vytvárajú rôzne poznámky, pričom často ide o rôzne úlohy. Autor si týmto spôsobom napríklad poznačí, že práve vytvorený úsek bude potrebovať jeho pozornosť neskôr. Vzniká tak určitý záznam úlohy. Aby si však autor alebo aj iná osoba čítajúca kód túto poznámku všimla medzi ostatnými komentármi, používajú sa rôzne značky. Azda najznámejšia značka je „TODO“, ktorá predstavuje anglické slovné spojenie „to do“ (urobiť). Autori Storey a kolektív, ktorí skúmali rolu anotácií úloh [3], zistili na základe vlastného prieskumu zastúpenie rôznych značiek u softvérových vývojárov. Prehľad uvádzam v Tab. 1.

Tab. 1. Zastúpenie používania značiek vývojového prostredia Eclipse u softvérových vývojárov [3].

Značka	Počet respondentov (N = 79)
TODO („urobiť“)	77 (98 %)
FIXME („oprav ma“)	34 (43 %)
XXX	12 (15 %)
OTHER („ostatné“)	11 (14 %)
HACK („rýchla oprava“)	6 (8 %)

Tieto anotácie úloh sa od úloh vedených v externých systémoch na sledovanie úloh odlišujú nielen svojim umiestnením. Úlohy zaznamenané takýmto spôsobom majú špecifické vlastnosti. Podľa mojich pozorovaní počas vlastnej práce na tímovom softvérovom projekte som zaznamenal, že komentáre a teda anotácie úloh sú vyjadrené často menej formálne než záznamy v na to určených nástrojoch, napríklad v systémoch na zaznamenávanie chýb. Rovnakú vlastnosť opisujú Ying a kolektív [4], ktorí sa tiež zaoberali prieskumom komentárov obsahujúcich úlohy. Dokonca zaznamenali mnoho anotácií obsahujúcich len jedno slovo alebo čiastočné vety. Tiež uvádzajú, že túto neformálnosť je možné vysvetliť na základe toho, že tieto komentáre majú často slúžiť len ako osobné pripomienky pre jednotlivých členov tímu. Moje skúsenosti toto vysvetlenie potvrdzujú. Vzhľadom na uvažovanie malého tímu by som však doplnil aj možnosť, že členovia tímu sa jednoducho bližšie poznajú a preto vytvárajú neformálnejšie komentáre nielen pre vlastnú potrebu, ale nie sú prekážky použiť ich aj pri zdieľaní v rámci tímu.

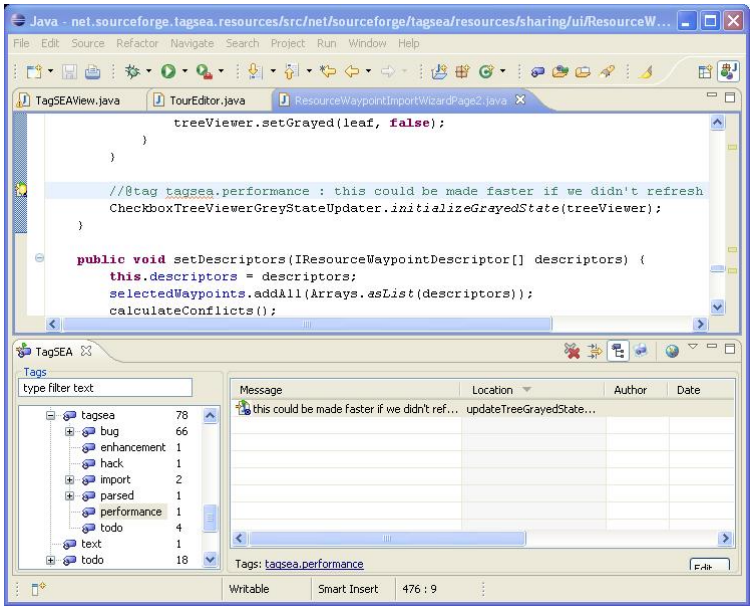
Ďalším dôležitým aspektom, v ktorom sa anotácie úloh odlišujú sú okolnosti, za ktorých vznikajú. Častý spôsob vzniku anotácií označených „TODO“ vyplýva zo spôsobu, ktorým pracujú ľudské bytosti. V prípade, že sústredený človek momentálne súvisle pracuje na určitej úlohe a narazí na problém alebo čiastkovú nevyriešenú úlohu, je pre neho jednoduchšie pokračovať na vyššej úrovni v rozpracovanej sústredenej činnosti a neskôr sa vrátiť k týmto menším častiam a sústrediť sa na ne zvlášť. V projektoch, ktorých som sa zúčastnil, bola značka „TODO“ takmer výhradne používaná práve takýmto spôsobom. Domnievam sa, že istú úlohu zohráva aj psychologický faktor, keď úlohy alebo chyby vložené do systému na sledovanie sú ihneď dostupné ostatným, pokiaľ v kóde a komentároch sa môže autor ľubovoľne navracať a upravovať ich, až kým nezanesie svoj kód do úložiska softvéru na riadenie verzií. Vzniká tak aj priestor na už

spomínané vlastné poznámky, ktoré samozrejme nemá zmysel zapisovať do externých podporných nástrojov na riadenie projektu. Ak však autor takéto svoje poznámky vyrieši alebo odstráni pred uložením kódu do úložiska, nie je potrebné sa týmito osobnými anotáciami zaoberať z pohľadu riadenia projektu a tímovej práce (a tým aj tejto eseje), pretože ich nikto iný ako dotyčný autor nikdy neuvidí.

Už naznačené rozdielne umiestnenie anotácií úloh v kóde od umiestnenia zápisov úloh vedených v externých systémoch na sledovanie úloh má tiež rôzne výhody a nevýhody. Výhodou anotácií v zdrojovom kóde je, že sú dostupné priamo na mieste, ku ktorému sa vzťahujú a to bez ohľadu na vývojové prostredie. Naopak na druhú stranu, pokiaľ vývojové prostredie neposkytne pre tento účel podporu, získanie prehľadu o úlohách zapísaných v zdrojovom kóde môže byť problematické. Ale tým, že tieto anotácie nie sú toľko „na očiach", môžu mať podľa mňa autori väčší pocit voľnosti pri ich vytváraní.

Podporné nástroje pre komentáre a anotácie úloh

Pre rôzne spôsoby použitia komentárov vznikli podporné nástroje. Pre dokumentačné komentáre existujú systémy umožňujúce automatické generovanie dokumentácie z komentárov v predpísanom formáte. Príkladom je nástroj Javadoc pre jazyk Java. Kým tieto nástroje do určitej miery podporujú úlohy (napríklad rozpoznanie značky „TODO" a vytvorenie prislúchajúceho výstupu), na účely podpory riadenia úloh nie sú priamo určené.



Obr. 1. TagSEA – pokročilý podporný nástroj pre anotáciu úloh [2].

Nástroje, ktoré bývajú integrované do vývojového prostredia, môžu mať význam v manažmente úloh. Napríklad prostredie NetBeans, ktoré je v kombinácii s jazykom Ruby obľúbené pri agilnom vývoji, vyhľadá v rámci načítaného projektu úlohy, čiže komentáre

obsahujúce značku „TODO“ a zobrazí ich zoznam spolu s informáciou o ich umiestnení. Pre prostredie Eclipse vytvorili autori skôr uvedenej štúdie anotácií úloh [3] pokročilejší nástroj TagSEA [2]. Nástroj podporuje okrem anotácií úloh aj mnoho ďalších značiek (Obr. **1**) a obsahuje ďalšie funkcie. Takéto nástroje však podľa môjho názoru slúžia skôr ako pomôcka uľahčujúca prehľad a prácu s anotáciami úloh a majú význam najmä na úrovni práce samotného člena tímu. Je zrejmé, že pri procesoch riadenia projektu na vyššej úrovni je ich priame využitie obmedzené (stále však môžu poslúžiť ako pomôcka).

V súvislosti s vývojovými prostrediami je nutné spomenúť možnosť automatického generovania úloh. „Tradičným“ využitím je vygenerovanie prázdnej metódy triedy prostredím. Do tejto metódy prostredie následne automaticky doplní komentár so značkou „TODO“ a poznámkou, že túto metódu je potrebné implementovať. Na druhú stranu, často je tento komentár príliš jednoduchý a obsahuje iba túto poznámku, pričom pre efektívne využitie anotácií úloh by bolo vhodné, aby obsahoval aj ďalšie údaje, napríklad časovú značku a identifikátor používateľa (člena tímu), ktorý túto anotáciu vygeneroval. Rovnako by tieto údaje pomohli aj pri ručne vytváraných anotáciách. Ak však prostredie priamu podporu pre podpisovanie komentárov nemá, doplnenie týchto údajov pomocou skriptovania v prostredí alebo externých programov nebude zložité.

Od anotácií úloh k metóde Scrum a riadeniu projektu

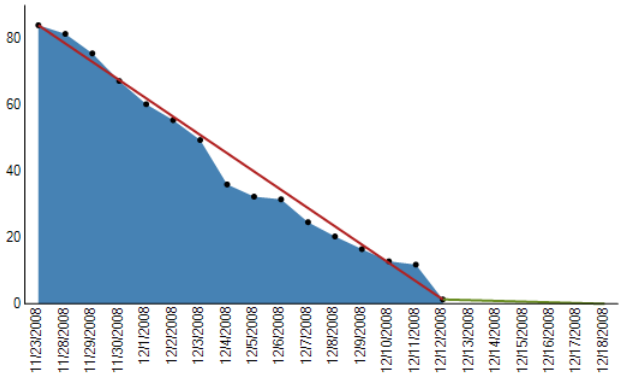
Pretože podaním obsiahleho návodu k agilným spôsobom vývoja a jednotlivým metódam by došlo k odchýleniu sa od zamerania eseje, poskytnem v tejto časti len zjednodušený prehľad metódy Scrum. Tento základný prehľad postačuje na pochopenie role anotácie úloh pri vývoji metódou Scrum.

Metóda Scrum predstavuje agilný spôsob vývoja softvéru. Táto metóda je iteratívna, vývoj je rozdelený do iterácií, ktoré sa nazývajú šprinty [1]. Zo zoznamu požadovaných funkcií (z angl. „backlog“) sa do šprintov vyberajú úlohy, za ktoré sú zodpovední jednotliví členovia tímu. V tejto metóde sa kladie veľký dôraz na sledovanie úloh a celkové napredovanie v rámci šprintov. Konajú sa tak denné stretnutia, kde každý člen tímu ústne podá krátku správu o tom, čo v daný deň na projekte vykonával, akú časť z aktuálnej práce už splnil, čo bude vykonávať do ďalšieho stretnutia a tiež či sa stretol s problémami, pri ktorých bude potrebovať pomoc ostatných členov tímu. Špecifickým nástrojom je tzv. diagram spálenia (z angl. „burn down chart“). Takýto diagram zobrazuje v kontexte plynúceho času ostávajúcu prácu, ktorú je potrebné vykonať v rámci aktuálneho šprintu a porovnáva ju s ideálnym priebehom. Príklad je uvedený na Obr. 2, kde červená čiara predstavuje ideálny priebeh a modrá oblasť vyznačuje ostávajúcu prácu. Z takéhoto diagramu v ľubovoľnom okamihu vidno, ako šprint prebieha a či je potrebné zrýchliť, alebo práca na projekte prebieha rýchlejšie ako sa predpokladalo a je možné pribrať ďalšie úlohy.

Z uvedených vlastností vyplýva, že presné sledovanie napredovania úloh je v metóde Scrum veľmi dôležité a nevyhnutné. Samozrejme existujú rôzne podporné nástroje, ako príklad uvediem RedMine alebo Agilo. Nie je mi však známe, že by takéto prostriedky pracovali s anotáciami v zdrojovom kóde. Ako som skôr opísal, anotácie úloh priamo v zdrojovom kóde vznikajú často vo väčšej rýchlosti a pri menších úlohách alebo problémoch. Práve táto vlastnosť anotácií sa podobá na dynamický až akčný ráz metódy

Scrum a zapadá do denného vyhodnocovania úloh, preto by prepojenie anotácií úloh v zdrojovom kóde s nástrojmi pre riadenie projektu prinieslo podľa mňa prínosy pre efektívnejšiu aplikáciu metódy Scrum. Z hľadiska podkladu pre realizáciu takéhoto prepojenia je situácia jednoduchá, úložiská zdrojového kódu je možné integrovať s nástrojmi pre podporu vývoja metódou Scrum.

Je však možné namietať, že niekto počas vývoja vytvorí viac anotácií budúcich úloh a pri tom dokončí väčšiu časť aktuálnej úlohy, ako niekto, kto vytvára úloh málo (odlišná jemnosť vytváraných čiastkových úloh u rôznych ľudí). Tu by bol priestor na aplikáciu personalizácie pre jednotlivých členov tímu, kde by sa systém pri odhadovaní napredovania v úlohe prispôsobil štýlu, akým vytvára anotácie úloh konkrétny člen tímu. Tento postup je však pomerne náročný, alternatívnym riešením by bolo rozšíriť údaje v anotáciách okrem skôr zdôvodneného času a identifikátora používateľa aj o údaj, v ktorom používateľ odhadne náročnosť úlohy. Bola by tak potrebná disciplína zo strany vývojárov pri tvorbe komentárov alebo podpora na strane vývojového prostredia. Niektoré údaje je tiež možné získať z úložiska zdrojových kódov (komentár vytvoril používateľ, ktorý ho zaniesol do úložiska a podobne), ale opäť by šlo o komplikovanejšie riešenie.



Obr. 2. Príklad diagramu spálenia (prevzaté z http://shipsoftwareontime.com).

Záver

Domnievam sa a na základe mojich úvah v tejto eseji si dovolím tvrdiť, že prostriedok anotácií úloh v zdrojovom kóde je v súčasnosti zo strany podporných prostriedkov pre riadenie softvérových projektov nedocenený. Ako som ukázal, vytváranie takýchto anotácií je pre softvérových vývojárov prirodzená záležitosť, ktorú už väčšina z nich vykonáva bežne. Charakter týchto anotácií je taký, že často ide o rôzne čiastkové problémy a úlohy vzniknuté pri riešení rozdelených úloh, čiže o zjemnenie pôvodných úloh. Preto napríklad v agilnej metóde Scrum, kde je dôležité sledovanie úloh, by sledovanie vývoja anotácií úloh v projekte mohlo poskytnúť cennú spätnú väzbu o podrobnejšom stave napĺňania aktuálneho šprintu.

V súčasnosti existujú nástroje, ktoré podporujú anotácie úloh v zdrojovom kóde na rôznej úrovni a pre rôzne účely. Samotný princíp, ako sú anotácie úloh používané na

riadenie úloh, podľa autorov Storey a kolektív však nie je často cieľom akademických výskumov, uvádzajú, že okrem ich práce [3] a práce Yinga a kolektívu [4] im nie je známy iný výskum priamo tejto oblasti. Výskum v príbuzných oblastiach (používanie a riadenie komentárov v softvérovom inžinierstve, podporné nástroje pre riadenie úloh) a pokračovanie výskumu autorov Storey a kolektív však indikuje, že o túto oblasť je záujem. Predpokladám teda, že v oblasti integrácie a prepojenia nástrojov na podporu anotácií úloh dôjde na základe aktívneho výskumu k posunu, ktorý aj odporúčajú Storey a kolektív.

Okrem toho, v komentároch v zdrojových kódoch sa okrem anotácií úloh nachádzajú aj ďalšie položky hodné skúmania v súvislosti s riadením softvérových projektov. Autori Ying a kolektív opísali komunikáciu medzi členmi tímu prostredníctvom komentárov a anotácií [4]. Táto téma je zaujímavá ako z hľadiska manažmentu ľudských zdrojov a podporných prostriedkov pre tento manažment, tak aj komunikácie v tíme, no presahuje rozsah tejto eseje a preto ju ponechávam len ako inšpiráciu k ďalším úvahám.

Použitá literatúra

1. Hu, Z., Yuan, Q., Zhang, X.: Research on Agile Project Management with Scrum Method. In: *IITA International Conference on Services Science, Management and Engineering*, IEEE Computer Society, Washington, DC (2009), 26-29.
2. Storey, M. a kol.: TagSEA 0.6.6 - Tags for Software Engineering Activities in Eclipse. Dostupné na internete: http://tagsea.sourceforge.net/research.html, [cit: 2009-Október]
3. Storey, M., Ryall, J., Bull, R., Myers, D., Singer, J.: TODO or to bug: exploring how task annotations play a role in the work practices of software developers. In: *Proc. of the 30th international Conference on Software Engineering*, ACM, New York (2008), 251-260.
4. Ying, A. T., Wright, J. L., Abrams, S.: Source code that talks: an exploration of Eclipse task comments and their implication to repository mining. In: *Proc. of the 2005 international Workshop on Mining Software Repositories*, ACM, New York (2005), 1-5.

Annotation

From Task Annotations in Comments to Project Management: Task Tracking

Task tracking is an inevitable aspect of software projects and therefore many support means exist. However there are situations when members of a software project development team do not turn to external tools, but because of various reasons they create task notes directly in source code in the form of comments. New task tracking support means is created in a natural way – task annotations in a source code. Support tools for these new means were created afterwards. In this essay I present my thoughts on circumstances where developers prefer annotations instead of available external tools, which are even already used in their project. Based on this reasoning, I weigh the role of task annotations in the teamwork and so the role, which these means take in a project management. When considering software projects, this essay aims at small teams that are using agile software development by the implementation of Scrum method.