PREČO MUSÍME KOMENTOVAŤ ZDROJOVÝ KÓD?

Tvorba softvérovej dokumentácie patrí medzi základné inžinierske činnosti rozhodujúce pre efektívny vývoj a kvalitu softvéru.

Dominik Rerko

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

Abstrakt. *Dokumentácia je súčasťou mnohých inžinierskych diel, softvérový produkt nevynímajúc. Väčšina ľudí ju síce považuje za nepotrebnú, avšak je rovnako dôležitá ako softvér samotný. Jej hlavnou úlohou je špecifikovanie určitého správania a činností pre ostatných v projekte zainteresovaných. Táto esej sa pokúša vysvetliť, prečo je komentovanie zdrojových kódov také dôležité a ako je to v skutočnosti. Čitateľovi nepredkladá jednostranný pohľad, snaží sa rozobrať problém z pohľadu rôznych aspektov. Taktiež porovnáva prístup tradičných metód, používaných pri vývoji softvéru s prístupom agilného, čo sa dokumentácie týka. V neposlednom rade sa pozerá na softvérovú dokumentáciu z manažérskeho hľadiska, kde rozoberá súvis medzi dokumentáciou a úspechom projektu.*

Kľúčové slová: *komentovanie zdrojového kódu, agilná dokumentácia, vplyv dokumentácie na úspešnosť projektu*

Úvod

Buďme k sebe úprimní. Nikto o nej nechce písať, čítať ju, či nedajbože podieľať sa na jej samotnom vytváraní. Možno si kladiete otázku, o čom to práve hovorím. Tí skúsenejší určite tušia, že sa jedná o dokumentáciu všeobecne považovanú za akúsi sekundárnu, resp. menej podstatnú časť diela. Pritom vôbec nemusí ísť len o softvérovú dokumentáciu. Množstvo ľudí sa značne znechutí už pri prvotnom vyslovení tohto slova a vôbec sa im

nečudujem. Veď kto by chcel celý čas len tak „tupo“ sedieť a písať tony mnohokrát „zbytočnej“ dokumentácie.

Pozícia softvérových projektov je v ohľade „zbytočnosti“ trošku iná a má značné špecifiká, keďže samotný softvér je nehmatateľný a proces jeho tvorby je úplne odlišný od iných produktov (napr. v stavebníctve). Existuje rozdielny pohľad na softvérovú dokumentáciu vyplývajúci z pozície pracovníka v IT firme. Pre manažéra alebo vedúceho projektu je kritická z pohľadu úspešnosti projektu, avšak pre ostatných to môže byť len nepríjemné obťažovanie odvádzajúce od skutočnej práce.

Softvérová dokumentácia pozostáva z viacerých častí, ako napr. technická, používateľská, marketingová časť a mnoho iných. Určite všetky majú zmysel a opodstatnenie, a ak by sme chceli dopodrobna špecifikovať a rozobrať každú časť, tak rozsah eseje by bol značne nad rámec a z tohto dôvodu som sa rozhodol bližšie venovať jednej časti technickej dokumentácie – komentovanie zdrojových kódov. V nasledujúcich častiach eseje sa pokúsim vysvetliť, prečo je dôležité komentovať zdrojový kód, aký súvis s dokumentáciou má zmena kódu, ako sa k dokumentovaniu stavajú pri vývoji softvéru agilnou metódou a v neposlednom rade, aký je súvis medzi dokumentáciou a úspechom projektu.

Prečo dokumentovať zdrojový kód?

Tak na túto otázku je veľmi ťažké jednoznačne odpovedať, avšak aj napriek tomu sa o to pokúsim. Predstavme si modelovú situáciu. Ako programátor potrebujem použiť externú knižnicu, ale nemám k nej dostupnú dokumentáciu. V lepšom prípade strávim zbytočné hodiny premýšľaním nad tým, čo tá-ktorá funkcia robí a nakoniec sa mi to nejakým spôsobom podarí. V tom horšom to odignorujete, a budete nútení urobiť si vlastnú, čo v konečnom dôsledku znamená značné množstvo vynaloženého úsilia úplne zbytočne.

Keďže softvér sa v drvivej väčšine vyvíja v tímoch, v tom prípade existuje spolupráca nielen medzi programátormi, ale aj v rámci kódu, je potrebné, aby všetci programátori vedeli, čo konkrétne tá-ktorá časť kódu robí [1].

Poniektorí by mohli namietať a argumentovať tým, že komentovanie bude viac na škodu ako na úžitok. Programátori nie sú naplno vyťažení ich prvoradou robotou (programovaním) a všetok ďalší čas, ktorý je potrebný na vytváranie dokumentácie, je pre nich zbytočným časom.

Ako je to v skutočnosti?

Úloha komentovania zdrojového kódu je často zanedbaná aj napriek tomu, že každý, kto píše zdrojový kód vie, akú pridanú hodnotu v skutočnosti prináša [1]. Čítanie kódu je podstatná časť softvérového inžinierstva a v praxi je kód oveľa viac čítaný, ako písaný. Komentáre umožňujú rýchlejšie a hlbšie porozumenie a v neposlednom rade zvyšujú úroveň čitateľnosti.

Tvorba softvérovej dokumentácie je základná inžinierska činnosť rozhodujúca pre efektívny a kvalitný vývoj softvéru. Bez ohľadu na zámer autora je všetok zdrojový kód tak, či onak znovu použitý, či už priamo, alebo iba so zámerom porozumieť mu [2]. V oboch prípadoch dokumentácia slúži na špecifikovanie správania pre ostatných zainteresovaných. Bez dokumentácie sme nútení získať potrebné informácie uskutočnením

nebezpečných pokusov, skúmaním implementácie, alebo v najhoršom prípade vysvetlením od samotného autora. Tieto alternatívy sú vo väčšine prípadov neprijateľné. Hoci sa niektorí vývojári domnievajú, že kvalitný zdrojový kód dokumentuje sám seba dostatočne, existuje veľké množstvo informácií o správaní kódu, ktoré nemôžu byť podané priamo z kódu bez akejkoľvek dodatočnej dokumentácie, pretože to vyžaduje silu a flexibilitu prirodzeného jazyka. Z tohto dôvodu si dovolím tvrdiť, že dokumentácia zdrojového kódu je nenahraditeľná nutnosť, rovnako dôležitá ako samotná disciplína. V konečnom dôsledku pomáha zvýšiť efektivitu a kvalitu vývoja akýchkoľvek softvérových produktov.

Zmena kódu, zmena dokumentácie

Ako určite všetci vieme, žiaden softvér nie je dielo na niekoľko desaťročí a skôr, či neskôr je potrebná určitá modifikácia, čo znamená zmenu kódu. Položme si jednoduché otázky. Mení sa spolu s kódom aj dokumentácia? Nie je dokumentácia „zastaraná“? Kedy zmena nastáva? Počas, alebo po úprave zdrojového kódu?

Výsledky prieskumu [1] naznačujú nasledovné:

1. Novo-pridaný kód, napriek tomu, že jeho množstvo rastie, je sotva zdokumentovaný.
2. Deklarácie tried a metód sú komentované oveľa častejšie, ako napríklad volanie metódy.
3. 97 % všetkých zmien dokumentácie je spravených v rámci tej istej revízie, ako pridružená zmena zdrojového kódu.

Zarážajúci je samotný fakt, že nový kód zriedkakedy dostane adekvátne okomentovanie. O tom, prečo je tomu tak, by sa dalo polemizovať. Ani autori prieskumu nevedia nájsť jednoznačnú príčinu tohto problému [1]. Po dôkladnejšom uvážení som prišiel k názoru, že najpravdepodobnejším dôvodom samotného faktu je lenivosť programátorov. Vysvetľujem si to tým, že väčšina zmien v zdrojovom kóde spočíva len v malej obmene, teda programátorom sa nechce hľadať v komentároch, čo konkrétne zmenili. Prípadne majú dobrý pocit, že samotný komentár existuje, následkom čoho ich nehryzie svedomie.

Agilný vývoj a dokumentovanie

V posledných rokoch prechádza mnoho firiem z klasického inkrementálno-iteratívneho vývoja na agilné metódy. V takomto prípade sa kladie dôraz hlavne na samotný výsledok (fungujúci softvér). Agilný vývoj sa vyhýba činnostiam, ktoré priamo neprispievajú k úspechu projektu [3]. V kontexte vývoja softvéru agilnými metódami patrí medzi takéto činnosti napríklad dokumentácia, zohrávajúca až druhotnú rolu [3]. Dôvod, prečo je tomu tak, je pravdepodobne nasledovný.

Preferencia zdrojového kódu pred dokumentáciou si z môjho pohľadu pravdivo zakladá na fakte, že je efektívnejšie pracovať na niečom, čo je konkrétne, čo funguje a čo môže byť objektívne posudzované a modifikované, ako posudzovať a meniť niečo, čo je iba plán. Tento fakt však argumentuje iba proti určitej kategórii dokumentácie. Jedná sa o všetky objemné a podrobné dokumenty opisujúce plánovanú implementáciu, vytvorené

ešte predtým, než je problém dostatočne analyzovaný a pochopený. Dokumenty tohto typu rýchlo „zostarnú“ a veľká časť úsilia vynaloženého do výroby a ich aktualizácie je zbytočná. Napriek tomu, „neúčinnosť“ takýchto dokumentov nie je a ani nemôže byť jednoznačným argumentom na elimináciu dokumentácie, dokonca ani na jej vytváraní povrchne.

Podľa [3] nie je správne dokumentovať softvér na úrovni zdrojového kódu, kam komentáre zdrojových kódov nepochybne patria. Odôvodňujú to zbytočnosťou snažiť sa udržiavať duplicitné informácie konzistentné. Namiesto toho uprednostňujú dokumentáciu na vyššej úrovni abstrakcie, zbavenú „zbytočných” technologických detailov a požiadaviek. Tento prístup je prinajmenšom zaujímavý a v porovnaní s vývojom softvéru klasickými metódami značne v kontraste. Osobne sa viac prikláňam k alternatíve komentovania zdrojových kódov, čo sa pokúsim adekvátne zdôvodniť. Veď načo by nám bol softvér, ktorý by síce fungoval, avšak skôr či neskôr, keď príde moment zmeny, by sme neboli schopní zmenu vykonať z dôvodu chýbajúcej dokumentácie? Pravdepodobnosť, že takýto stav nastane je takmer 100 percentná. Ak by v takomto prípade nebola dokumentácia zdrojových kódov k dispozícii, programátori by prinajmenšom „bojovali“ s vylepšením, prípadne s opravou softvéru, čo by značne predĺžilo termín realizácie a samozrejme cenu diela.

Aký je vzťah medzi dokumentáciou a úspechom projektu?

Nateraz sa skúsme odosobniť od roly programátora a vžime sa do funkcie manažéra. Z podstaty pozície manažéra nebude dôležité či programátor strávi nejaký čas navyše tvorbou dokumentácie, alebo či je tá- ktorá metóda okomentovaná. Jeho prvoradý záujem spadá do oblasti úspešnosti celého projektu, spolu s úzko súvisiacim ziskom.

Ak si myslíte, že existuje spoľahlivý vzťah medzi úspechom projektu a písaním obsiahlej dokumentácie, tak vás asi sklamem. V skutočnosti je viac pravdepodobné, že čím viac dokumentácie napíšete (vrátane dokumentácie zdrojových kódov), tým je väčšia šanca na neúspech projektu [4].

Zažili ste už projektový tím, ktorý strávil drahocenný čas písaním vyčerpávajúcich požiadaviek, dal ho podpísať zákazníkovi len preto, aby vývojári mohli vytvárať stále niečo iné? Alebo tím špecifikujúci požiadavky len, aby mu zákazník povedal, že to v skutočnosti nie je to, čo chce? Alebo vývojára ignorujúceho vytvorený model, ktorý neskôr zahodí a nakoniec programuje taký, ako chce? Osobne si neviem predstaviť manažment, ktorý by takéto niečo dopustil. V konečnom dôsledku by to znamenalo absolútny neúspech projektu a namiesto produktu by sme mohli zákazníkovi odovzdať nepotrebnú dokumentáciu.

Prečo sa potom ľudia mylne domnievajú, že dokumentácia je rozhodujúcim faktorom úspechu vo vývoji softvéru? Myslím si, že odpoveď na túto otázku spočíva v prechode IT oddelení z vývoja typu „naprogramuj a oprav“ na vývoj vodopádový s veľkým množstvom dokumentácie, ktorý nastal v 80-tych rokoch minulého storočia. Táto zmena zlepšenie skutočne priniesla [4]. Podobné výsledky je dnes možné vidieť s CMMI. Pri prechode z úrovne 1, na úroveň 2, alebo 3 v skutočnosti bádať zlepšenie v produktivite. Skrýva sa za tým neskutočné množstvo dokumentácie. Teraz si možno hovoríte, že si protirečím a samotný fakt potvrdzujem, avšak nie je to pravda. Zistením tejto informácie

sa ľudia v tomto fakte utvrdili a s odpoveďou sa uspokojili, čo môžeme považovať za chybu. Zástancovia CMMI alebo iných stratégií, ktoré nevyhnutne obsahujú kvantum dokumentácie, si pravdepodobne nikdy nepoložia otázku, či neexistujú lepšie spôsoby, ako vytvárať softvér. Neprinesie zameranie úsilia na tvorbu vysoko kvalitného softvéru väčšiu návratnosť, ako písanie dokumentácie? Neprinieslo by nájdenie spôsobu, ako vytvárať kód vysokej kvality (napr. refaktorovanie) väčšiu návratnosť, ako písanie dokumentácie? Myslím si, že pravé hodnota dokumentov nie je dokumentovanie samého seba. Ich prínosom je zlepšenie pochopenia problémovej domény, čo môžeme dosiahnuť zameraním sa práve na modelovanie úsilia (napr. agilný vývoj), dosiahnuť tak vyššiu kvalitu softvéru a nemalou mierou si tým zvýšiť šancu na úspech projektu.

Záver

Na záver by som rád zdôraznil, že kvalitná a v rozumnej miere použitá dokumentácia zdrojového kódu je jedným z hlavných predpokladov na kvalitný softvérový produkt. Aj keď sa to na prvý pohľad nezdá, v konečnom dôsledku dokáže ľudom usporiť množstvo času a úsilia, a tým aj cenu produktu.

Použitá literatúra

1. Fluri, B., Wursch, M., Gall, H.C.: Do Code and Comments Co-Evolve? On the Ralation between Source Code and Comment Changes. In: *Reverse Engineering, 2007. WCRE 2007. 14th Working Conference on,* vol., no., pp.70-79, 28-31 Oct. 2007
2. Remco C. de Boer, Hans van Vliet.: Writing and Reading Software Documentation, 2009
3. Christoph, J.S, Werner,H.: Necessary and neglected? An empirical study of internal documentation in agile software development teams. In: *Proceedings of the 29th ACM international conference on Design of communucation*, pp. 159-166, USA, New York 2011
4. Selic, B.: Agile Documentation, anyone? *IEEE Software*, vol. 26, no. 6, Dec. 2009

Annotation

Why do we have to comment source code?

This essay attempts to explain, why commenting of source code is so important. It does not present a one-sided view, it tries to break down the problem in terms of different aspects. It also compares documentation approach to traditional methods used in software development and agile approach. Finally, it looks at the management view of software documentation, where discusses the relation between documentation and project success.