SÚ KOMENTÁRE V ZDROJOVOM KÓDE NAOZAJ POTREBNÉ?

Všetko s mierou.

Ľuboš Gelányi

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
gelanyi08[zavináč]student.fiit.stuba[.]sk

Abstrakt. *Vzhľadom na súčasné narastajúce nároky na softvérové projekty úmerne rastú aj požiadavky na kvalitný a správne dokumentovaný zdrojový kód. Metódami dokumentácie tohto softvérového artefaktu sa zaoberajú rôzne techniky a postupy. Paradoxne tie najviac zaužívané prinášajú do zdrojového kódu elementy, ktoré by sa v ňom nemali nachádzať. Komentáre, teda text v zdrojovom kóde bez inej ako len informačnej funkcionality sú tradične zaužívane pri generovaní dokumentácie. Prinášajú však rôzne komplikácie, ktorým sa však za istých okolností je možné vyhnúť. Otázka však zostáva či komentáre v zdrojovom kóde sú relevantné alebo nie.*

Kľúčové slová: *softvérová dokumentácia, komentáre, generátor, zdrojový kód*

Úvod

V prvom rade je potrebné si položiť otázku aký ma vytváranie softvérovej dokumentácie význam. Z programátorského hľadiska je dokumentácia dôležitá pre porozumenie kódu, ktorý bol vytvorený iným členom tímu. Len takto je možné dodržať zlaté programátorské pravidlo DRY (don't repeat yourself), čo vo voľnom preklade znamená neopakuj svoju prácu [2]. Správnym poznaním zdrojového kódu, ktoré nám ponúkne práve dokumentácia sa minimalizuje riziko vytvárania duplicitnej práce.

Jednou z možností ako vytvárať dokumentáciu zdrojového kódu je manuálne zapisovanie metód a algoritmov, prípadne celkových fragmentov zdrojového kódu. Táto metóda je však jednak náročná na čas ako aj v konečnom dôsledku značne neprehľadná.

Otázka však znie prečo vytvárať dokumentáciu do separátneho dokumentu. Isteže je potrebné dodržiavať zaužívané postupy a štandardy ale zrejme žiadny programátor nepohrdne možnosťou získať potrebné informácie priamo v súbore so zdrojovým kódom [2]. Toto je možné dosiahnuť doplňovaním textu vo forme komentárov priamo do zdrojového kódu. Je však vhodné vkladať do zdrojového kódu enormné množstvo z programátorského hľadiska "prázdneho" textu? Pre potreby generátorov softvérovej dokumentácie áno.

Každopádne ani dokumentácia generovaná na základe komentárov v zdrojovom kóde nie je v každom ohľade plne relevantná. Napríklad nepodporuje WYSIWYG prístup, či interaktívny obsah prepojený priamo so zdrojovým kódom [1]. Takto vytvorená dokumentácia je jednoznačným prínosom v rámci formálnych a štylistických úprav vytvorených dokumentov. Zrejme však nebude taká jednoduchá pre samotného programátora, ktorý musí upustiť od tradičného písania zdrojového kódu a prispôsobiť sa obohatenému obsahu. Výsledný prototyp na demonštrovanie metódy je taktiež implementovaný len do jedného vývojového prostredia.

Najjednoduchším riešením v tomto prípade vyplýva jednoduché generovanie dokumentácie zo zdrojového kódu. Je však dôležité si uvedomiť čo je dôležité dokumentovať. Za žiadnych okolností nie je vhodné dokumentovať tisícky riadkov kompletného projektu. V zdrojovom kóde je potrebné vypichnúť naozaj dôležité fragmenty kódu. Pre tento účel môžu byť využité rôzne techniky ako napríklad vyhľadávanie informácií v texte (information retrieval) alebo porovnávanie vzorov (pattern matching) [4]. Táto metóda si však vyžaduje značné vstupné úsilie.

Riešenie nastolenej problematiky sa líši aj v názoroch odborníkov je však isté, že ideálnu metódu nie je možné vytvoriť nakoľko, každý programátor má rozdielny vlastný postoj či očakávania a každému môže vyhovovať iná forma dokumentácie.

Dokumentácia v zdrojov kóde?

Patrí do zdrojového kódu aj nefunkcionálny text? Určite áno ale v rozumnej miere. Komentáre slúžia len ako doplnok k samotnému fragmentu a preto je potrebné s nimi narábať veľmi opatrne. Kód, ktorý potrebuje refaktorizáciu, je zlý alebo neprehľadný je potrebné opraviť či kompletne vymeniť a nie dopĺňať komentármi [2]. Je všeobecne známe, že dobrý kód je samoopisný a teda, že všetky potrebné informácie získa programátor priamo z funkcionálnej časti kódu a žiadne komentáre nie sú potrebné.

Napriek tomu sa však komentáre v zdrojovom kóde mohutne používajú. A napriek tomu, že komentárom by sme sa mali vyhýbať je ich používanie osožné. V rámci rozsiahlejších projektov pristupuje k tomu istému kódu viacero programátorov. Nie zriedka sa stáva, že programátor opustí pracovisko a jeho nasledovník práve vďaka vhodným komentárom dokáže pokračovať v jeho práci.

Stále však platí pravidlo písania kódu, ktorý sám dokáže vysvetliť svoju funkciu na základe vhodne zvolených názvov metód, atribútov či tried. Problém však nastáva pri generovaní dokumentácie. A to práve preto, že účel dokumentovania nespočíva len v

pochopení zdrojového kódu iným programátorom ale aj v archivácii dosiahnutých výsledkov. Takéto dokumenty dokážu jednoducho vytvárať generátory dokumentácie. Medzi najzákladnejšie z nich patria Doxygen a Javadoc [3]. Pre tieto nástroje je vyslovene nutné komentovať zdrojový kód aj keď sa nejedná o komentáre v pravom slova zmysle. Výhodou metódy generovania dokumentácie je, že nie je potrebné poznať základnú syntax použitého jazyka. Generátor neskúma samotný kód ale len vyhľadáva špeciálne značky a extrahuje informácie k nim prislúchajúce. Celý proces je založený na lexikálnej a syntaktickej analýze značiek obsiahnutých v komentároch [3] na základe, ktorých dokážu tieto nástroje generovať dokumentáciu.

Tento postup je štandardne zaužívaný aj keď značne obmedzujúci. Samotný autor je nútený naštudovať základnú notáciu pre tieto nástroje a dôsledne ju dodržiavať. Na druhej strane však tieto značky môžu nahradiť samotné komentáre a tým ponechať v zdrojovom kóde len spracovateľný text či už samotným kompilátorom alebo generátorom dokumentácie.

Dynamická dokumentácia

Dokumentácia generovaná na základe komentárov je určite veľkým prínosom pre dokumentaristov, ktorých prácu vlastne suplujú programátori. Vygenerovaný text je už následne možné spracovávať do rôznych typov dokumentov. Výrazné plus dodávajú týmto nástrojom aj ich doplnky, napríklad generovanie UML diagramov. Čo však v prípade, že potrebujeme niečo viac? Dobrá softvérová dokumentácia by mala obsahovať rôzne pohľady na systém či už statický, dynamický, externý či interný na rôzne úrovne abstrakcie použitím rôznych multimediálnych prvkov ako napríklad video, obrázky či samotný text [1].

Problematikou zdynamizovania softvérovej dokumentácie sa venujú softvéroví inžinieri z kanadskej Univerzity Concordia [1], ktorí vytvorili SE-editor ako doplnok priamo do vývojového prostredia schopný zobrazovať multimediálny webový obsah (rich web content). Riešenie je podporované pomocou Web 2.0. Táto metóda je postavená na spojení možností webového prehliadača a editora zdrojového kódu. Toto vzájomné obohatenie poskytuje samotnému programátorovi nový pohľad na zdrojový kód.

Rovnako ako Web 2.0 aplikácie dokážu podporovať sociálnu a kolaboratívnu povahu zdieľania informácií dokáže aj metóda a z nej vyplývajúci nástroj SE-editor dokáže poskytnúť viac kolaboratívne prostredie pre dokumentáciu [1].

Napriek tomu, že niektoré časti kódu je možné v tomto prostredí dokumentovať pomocou rôznych multimediálnych prvkov určite nedokážu nahradiť informačnú hodnotu vhodne sformulovaného textu. Niektoré triedy môžu byť implementované s takou zložitosťou, že pochopenie tejto funkcionality môže skonzumovať značné množstvo času. Preto má táto metóda síce výraznú pridanú hodnotu ale nerieši základné poslanie dokumentácie a to poskytnúť základnú informáciu o zapísanom kóde. Ako som už spomínal, prehľadné diagramy dokážu jednoducho generovať aj samotné nástroje pre generovanie dokumentácie. Nástroj Javadoc dokonca poskytuje možnosť úpravy textu v komentároch pomocou HTML tag-ov, ktorými je možné text formátovať [3]. Aj keď myšlienka pridania kompletného portfólia multimediálnych prvkov do dokumentácie je

určite zaujímavá výsledná forma neposlúži ako vhodný dokument podľa zaužívaných postupov.

Analýza zdrojového kódu

Ako som už spomenul v úvode, je veľmi dôležité si uvedomiť, ktoré fragmenty kódu je potrebné dokumentovať. Jednoduchým riešením môže byť potrebné informácie vyhľadať. Problematikou sa zaoberajú metódy vyhľadávania informácií či porovnávania vzorov.

Výskumník Zanoni so svojím tímom [4] sa zaoberá využitím týchto metód pre zlepšenie výstupov nástrojov pre dokumentovanie zdrojového kódu. Celková metóda spočíva v analýze zdrojového kódu. V štruktúrovanom texte, ktorým zdrojový kód určite musí byť sa vyhľadávajú základné artefakty. V značnej miere sa pre tieto účely využíva porovnávanie vzorov [4]. Vo väčšine prípadov sa ako porovnávací vzor používa slovo. V tomto prípade by však metóda mohla byť obohatená aj o iné vyhľadávacie vzory ako sú napríklad celé definície tried, metód či stavov. Takýmto spôsobom by mohol byť zdrojový kód značkovaný len v potrebných prípadoch čo by značne minimalizovalo redundantné informácie.

Skúmaná metóda však nevytvorí dokumentáciu zdrojového kódu. Slúži len pre samotné generátory na identifikovanie relevantných častí v projekte. Je potrebné aby kód obsahoval potrebné komentáre napríklad pre nástroj Javadoc aby výsledky analýzy mohli byť spracovateľné práve samotným generátorom. Tu však vystupuje priestor na polemiku či sú značky obsiahnuté v týchto komentároch naozaj potrebné pri tejto metóde.

Tak písať alebo nepísať komentáre?

Pre potreby generovania dokumentácie sú určite dôležité. Môžu však spôsobiť neprehľadnosť kódu a taktiež znížiť produktivitu programátorov, ktorý na úkor funkčného kódu píšu komentáre.

V ideálnom prípade by komentáre boli v kóde prítomné ale nezaťažovali by programátorov až v takej značnej miere ako im to predkladajú pravidlá pre jednotlivé generátory. Ako vhodné riešenie sa javí využitie techník na analýzu zdrojového kódu a následnú extrakciu dôležitých fragmentov kódu. Pre takto získané bloky by autor už jednoducho mohol dopísať potrebné informácie pre generátory dokumentácie.

Základným identifikačným prvkom by sa pre samotné nástroje stali bloky kódu identifikované ako dôležité na dokumentovanie pomocou spomenutej metódy analýzy kódu [1] a k nim by sa priradili jednotlivé komentáre. Týmto spôsobom je možné ponechať programátorom ich zaužívané intencie písania komentárov ale na druhej strane ich neobmedzovať potrebnou notáciou pre generátory dokumentácie. Problémom ostáva zotrvanie komentárov v kóde, čo je mimochodom jeden z pachov vhodných pre refaktorizáciu. Pravdou však ostáva, že samoopisný kód je niekedy ťažký oriešok akokoľvek je jeho prínos relevantný [2]. Preto odporúčam písať zdrojový kód tak aby jeho funkcionalita bola pri prvom pohľade jasná. Ak sa to však nedá je určite vhodné fragmenty správne komentovať a zároveň tým dopomôcť dokumentovať samotný kód. Ak sa komentáre nachádzajú v zdrojových súboroch v rozumnej miere nie je možné aby sa kód stal neprehľadný. Práve naopak.

Záver

Hlavná myšlienka ostáva verná svojej jednoduchosti. Všetko s mierou. Komentáre by sa v zdrojovom kóde nemali nachádzať vôbec, je však jasné, že v niektorých prípadoch sú potrebné.

Polemizované metódy ukázali, že aj keď je možné sa komentárom vyhnúť nie vždy by to bolo na prospech. Najjednoduchšou formou dokumentovania zdrojového kódu je práve ich zozbieranie a vyhodnotenie čo ale prináša pre programátorov ďalšie bremeno v podobe nových štandardov, ktoré je potrebné nutne dodržiavať. Dynamická dokumentácia je síce oku lahodiace a príjemné riešenie, odkláňa však programátorov od základného zaužívaného modelu písania kódu. Úspech spočíva v identifikovaní podstatných fragmentov kódu načo nám poslúžia techniky vyhľadávania informácií.

Preto je vhodné spojiť príjemné s užitočným a to dovoliť programátorom komentovať zdrojový kód klasickým zaužívaným spôsobom a následne identifikovať objekty, metódy či premenné a následne komentáre k nim prislúchajúce. Generovaná dokumentácia má rovnakú kvalitu ako pomocou zaužívaných generátorov a pritom nijako nenarúša zaužívané postupy pri písaní zdrojového kódu.

Použitá literatúra

1. Schugerl, P., Rilling, J., Charland, P.: Beyond generated software documentation — A web 2.0 perspective. Software Maintenance, 2009. ICSM 2009. IEEE International Conference on . IEEE, 2009.
2. Spinnelis, D.: Code Documentation. Software, IEE. IEEE, 2010.
3. Van Deursen, A., Kuipers T.: Building documentation generators. Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on. IEEE, 1999.
4. Zanoni, J.C., Ramos, M.P., Tacla, C.A., Sato, G.Y., Paraiso, E.C.: A semi-automatic source code documentation method for small software development teams, CSCWD, 15th International Conference on. IEEE, 2011.

Annotation

Are source code comments really necessary?

This essay is dealing with source code comments necessity according to creating software documentation. Author evaluates different methods concerning approaches for generating source code documentation. Main issues are dealing with standard source code generators or advanced methods describing this process such as code analysis based on information retrieval or pattern matching and Web 2.0 perspective for software documentation. Whole problem is issuing through code comments which are disputed whether to use them or not.