VPLYV KVALITY DOKUMENTÁCIE NA TVORBU SOFTVÉROVÉHO PRODUKTU

Nie je dokumentácia ako dokumentácia.

Michal Igaz

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
michal.igaz@gmail.com

Abstrakt. *Esej vypovedá o problematike dokumentácie ako takej a o problémoch, ktoré sú späté s jej tvorbou. Vysvetľuje, ako sa dokumentácia stala silným nástrojom rôznych zložiek manažmentu projektu a tiež jej predajnosti klientovi. Subjektívne je tu opísaný význam a vplyv dokumentácie na projekt spolu s podrobnejším opisom týchto dvoch termínov. Autor využil svoje osobné skúsenosti s danou problematikou a vnáša do eseje aj svoje stanovisko na vplyv dokumentácie na projekt, jej význam pre rôzne zložky tímu, ktorý softvérový produkt vytvára a tiež aj tvorbu samotnej dokumentácie. Okrem toho vnáša do eseje aj kritiku nedostatku či prebytku dokumentácie a navrhuje alternatívy, ktoré by tento problém mohli pomôcť vyriešiť aj za cenu počiatočných prekážok.*

Kľúčové slová: *význam dokumentácie, vplyv dokumentácie, rozsah dokumentácie*

Úvod

V dávnejších časoch, v niektorých prípadoch ešte aj v súčasnosti, pomerne často zlyhávali celé zástupy softvérových projektov. Celé tímy odborníkov túto situáciu skúmali a snažili sa zistiť, čo je príčinou tohto javu a aké kroky je potrebné podniknúť pre zmiernenie jeho účinkov. Na konferenciách v rokoch 1968-1969 boli zavedené pojmy ako softvérová kríza a softvérové inžinierstvo [4]. Na týchto konferenciách boli identifikované hlavné problémy, s ktorými sa odborníci museli vysporiadať.

Jedným z podstatných problémov a tiež tým, ktorému sa budem v tejto práci venovať, je nedostatok dokumentácie. Nedostatok dokumentácie je najmä o nedostatku špecifikácie, či už funkčnej alebo technickej, ktorá ovplyvňuje najmä vývoj softvérového produktu a mieru jeho kvality, splnenia požiadaviek a iné [4]. Špecifikácie sú však súčasťou plánovania softvéru a tvoria ich prevažne analytici. Okrem toho existujú rôzne ďalšie typy dokumentácií, napríklad dokumentácia vývojárov, komentáre v kóde a tiež dokumentácia testovania a akceptačných testov.

V konečnom dôsledku nezáleží na type dokumentácie, lebo vo všeobecnosti sa dokumentácia počas vývoja softvéru mení. Zmeny v špecifikácii sa robia každú chvíľu v dôsledku nových prianí klienta, protestov programátorov a v niektorých prípadoch aj na podnet testerov. Zmeny v samotných technických dokumentáciách produktu sa dejú takmer vždy v dôsledku zmeny špecifikácie, prípadne vynútenou zmenou zdrojového kódu. No a zmeny akceptačných testov sú celkom bežné, najbežnejšie v záujme otestovania softvéru v čo možno najväčšej miere.

Význam dokumentácie

Nakoľko sa písanie dokumentácie ťahá počas celého vývoja softvéru a neodmysliteľne patrí k všetkým etapám životného cyklu softvéru, je zrejmé, že má mimoriadny význam. Tento štatút získavala postupne v priebehu vývoja tisícok softvérových produktov. Avšak nebolo to tak vždy. Pri už spomínanej softvérovej kríze bol jeden z identifikovaných problémov nedostatok dokumentácie [1]. Nebolo to len v tom, že by žiadna dokumentácia neexistovala, ale nebola to tá „správna“ dokumentácia. Ku každému projektu sa viac menej dôsledne vyvíjala aj dokumentácia, nebola však nijak ucelená a nemala definované žiadne pravidlá ani podstatné náležitosti. Dokumentáciu zväčša písali programátori, ktorí sa tejto nepríjemnej činnosti snažili čím skôr zbaviť a umlčať protesty svojich nadriadených. Takto vznikali niekedy aj viac ako tisíc stranové dokumentácie. Umlčali tým síce svojich nadriadených, ale ich zákazníci začali kričať oveľa hlasnejšie. Aký k tomu mali dôvod? Siahodlhé technické manuály a opisy algoritmov boli síce dobré pre ľudí, ktorí sa zaoberali vývojom, ale obyčajní priemerní používatelia sa v tej kope technických informácií strácali. Pokiaľ nebol softvér dostatočne intuitívny, nevedeli sa v ňom ani len orientovať, nevedeli a ani nechápali obmedzenia softvérového produktu, s ktorým pracovali. Nebolo im vôbec dodané to, čo potrebovali zo všetkého najviac: Používateľská príručka [2]. Takéto príručky boli v tých časoch veľmi vzácne. Boli síce písané, avšak ľuďmi, ktorí boli vrcholne atypickými používateľmi – programátormi. Používateľská príručka však bola iba jedným z viacerých kameňov úrazu medzi softvérovou firmou a klientom.

Aké boli tie ďalšie? Ďalším veľkým problémom sa stal predaj softvéru zákazníkovi. Pri získaní zákazky sa zišli zákazník a zástupca z firmy a vytvorili určitý koncept softvéru [1]. Zákazník nadiktoval, čo od softvéru očakáva a spokojný odišiel. Firma potom zadala úlohy svojim programátorom, ktorí sa pustili do práce a vytvorili plne funkčný robustný systém. Pri odovzdávaní a nasadení softvérového produktu u zákazníka však nastal problém. Otázka „čo má toto znamenať?“ sa stala nočnou morou každej softvérovej firmy, ktorá nekládla dostatočný dôraz na zákazníka. Ten len s hrôzou pozeral, aký softvér práve dostal do rúk a ako veľmi sa líši od toho, čo chcel. Išlo o jasné zlyhanie komunikácie. To, čo

takýmto nedorozumeniam bránilo, bola špecifikácia požiadaviek a časom aj funkčná špecifikácia riešenia. Špecifikácia požiadaviek sa stala hlavnou úlohou analytikov a konzultantov, aby tak zaistili dokonalé pochopenie klientových očakávaní [2]. Funkčná špecifikácia potom prikazovala programátorom, ako majú softvér vytvoriť, a nemali viac voľnú ruku pri výbere metód. Tieto opatrenia náramne zvýšili predajnosť softvérových produktov.

Vplyv dokumentácie

Dokumentácia má určite veľký význam a je jasné, prečo a ako tento význam získala. Táto znalosť je kľúčová pri identifikácii jednotlivých sfér vplyvu dokumentácie na softvérový produkt. Aké to vlastne sú? Dokumentácia ako taká vplýva na veľké množstvo vecí. Je to veľmi silný nástroj komunikácie v tíme, dôležitá súčasť zaúčania nových pracovníkov, nevyhnutná pre údržbu softvéru či ďalší vývoj [4]. Zaujímavé je, že aj napriek takémuto širokému spektru použiteľnosti dokumentácie bola v minulosti len niečo ako nepríjemná povinnosť.

Dokumentácia sa ako nástroj komunikácie používa už dlhší čas, nebola to však preferovaná forma. Bežnejšie bolo komunikovať osobne, znova a znova vysvetľovať svoje rozhodnutia a dizajn, možno časti kódu a iných, ako spísať to vo forme dokumentácie [3]. Pokiaľ však vedel tvorca časti systému spísať v dokumentácii svoje myšlienky bez ohľadu na to, či to bol analytik, vývojár, tester alebo manažér, ušetrilo to množstvo času na konzultáciách ohľadom projektu [4]. Programátor už nemusel neustále otravovať analytika, čo myslel tým či oným, tester nemusel otravovať programátora, že čo má vlastne ten algoritmus robiť. Nakoľko informatici sú ľudia s vysokou cenou práce, úspora času viacerých zamestnancov v konečnom dôsledku prinesie zisk. Táto motivácia sa však dá použiť aj pri ďalšom prípade.

Zaúčanie do systému je odjakživa chúlostivý proces. Ľudia, ktorí sú za daný program zodpovední, musia obetovať časť svojho pracovného času na zaúčanie nových spolupracovníkov. Dobrá dokumentácie tento problém neeliminuje, ale rapídne znižuje množstvo potrebného času [3]. Zaúčanie síce trvá rovnaký čas, ale zaúčaný nepotrebuje také množstvo času svojich kolegov, ktorých skúsenosti sa môžu využiť aj lepšie. Úspora nákladov rastie tým viac, čím viac má firma zamestnancov, pretože tam sa vystrieda väčšie množstvo ľudí.

Ďalší vývoj projektu je tiež oblasť, v ktorej príde vhod dobre napísaná dokumentácia. Do transparentného a dobre navrhnutého projektu sa rozhodne vkladajú ďalšie funkcionality lepšie ako do zle okomentovaného a zanedbaného [1]. Rovnako pri tvorbe nových inštancií už vytvoreného projektu je dokumentácia užitočná. Tento prípad je veľmi častý najmä v prípade softvérových projektov, ktoré sú univerzálne, avšak každý klient má určité špecifické požiadavky, napríklad vlastnú databázu [2].

Údržba softvéru z dokumentácie jednoznačne profituje. Pri riešení problémov z prevádzky je veľmi jednoduché otvoriť dokumentáciu a nájsť v nej sporné miesta. Na základe problémov a tiež na základe dokumentácie je možné navrhnúť rýchle riešenie chyby [3]. Toto by nebolo možné, keby pracovník najprv musel naštudovať funkcionalitu veľkej časti softvéru, kým by bol schopný odhadnúť, kde je chyba.

Dĺžka dokumentácie

Vplyv i význam dokumentácie by už mal byť jasný. Sú však aj prípady, kedy je množstvo dokumentácie príťažou. Kde sa nachádza správny medzník rozsahu dokumentácie? Vývoj dokumentácie nie je ani v tomto smere jednotný. Príliš mnoho dokumentácie škodí rovnako ako jej nedostatok [1]. Používateľ môže zostať frustrovaný pri vyhľadávaní informácie v tisíckach strán textu. Je to časovo neefektívne, prebúdza to negatívne pocity a tiež to má vplyv na zákazníkovu preferenciu zadania podobných projektov v budúcnosti. Veľké množstvo dokumentácie teda dokáže používateľa frustrovať, a nie reálne pomôcť. Oveľa vhodnejšia cesta je napísať menej výstižnej dokumentácie. Na takú prácu však človek musí mať nadanie. Snaha o vyjadrenie vlastných myšlienok stručne, jasne a výstižne môže byť mimoriadne náročná, tým viac, že na podobné veci zväčša nezostáva pri vytváraní softvérového produktu čas [5].

Dá sa však nejako skrátiť celkové množstvo dokumentácie bez toho, aby bola ovplyvnená jej opisnosť? Podobná otázka už napadla mnohým. Pojem, ktorý sa v súvislosti s týmto problémom objavil, sa nazýva „minimal documentation“, čo v preklade znamená minimálna dokumentácia [5]. Rozsah dokumentácie sa dá určite skrátiť použitím určitých tzv. „best practises“, čo v preklade znamená niečo ako osvedčené postupy. Určitou alternatívou je tvorba dvoch druhov dokumentácií, tzv. stručnej a komplexnej.

Ktoré sú tie osvedčené postupy? Ide hlavne o súbor techník, ktoré umožňujú tvorcovi dokumentácie pochopiť a stručne rozviesť svoje myšlienky pri písaní za zachovania jej malého rozsahu. Patria sem prevažne postupy písania či zásady jednoduchosti dokumentácie. Tie zahŕňajú hlavne všeobecné postupy, ktoré sa dajú využiť pri písaní dokumentácie tak, aby mala čo možno najvyššiu výpovednú hodnotu na čo možno najnižší obsah. Najdôležitejšou zásadou v tejto časti je s najväčšou pravdepodobnosťou využitie možnosti vygenerovania dokumentácie [5, 6]. Toto umožňuje vyťažiť maximum z práce programátorov za predpokladu, že sa riadili určitými konvenciami písania a komentovania zdrojového kódu. Medzi ďalšie zásady patrí nutnosť udržiavania jednoduchosti dokumentácie. Nesmie sa to však preháňať, ako píše Ambler: „don’t insult reader’s intelligence“ [6], čo v preklade znamená neurážaj čitateľovu inteligenciu. Minimálna dokumentácia sa dá dosiahnuť aj znížením celkovej redundancie. Prekrývanie sa jednotlivých častí je celkom bežné a dôsledne sa proti nemu bojuje. Veľkým problémom najmä pri agilnom vývoji softvéru sú rýchle a časté zmeny špecifikácií. Preto je dôležité vytvárať iba také diagramy alebo časti dokumentácie, pri ktorých je reálny predpoklad, že sa už nebudú nijak zvlášť meniť.

Ďalšou možnosťou písania krátkej a zmysluplnej dokumentácie je prístup vyvinutý pánom Donaldom Knuthom. Tento prístup sa dotýka problematiky tvorby dokumentácie viac okrajovo ako predchádzajúce osvedčené postupy. Vychádza z predpokladu, že programátor, ktorý tvorí zdrojový kód, je schopný explicitne opísať jeho funkčnosť [3]. Zároveň s kódom tvorí aj určitú esej, v ktorej opisuje funkčnosť programu. Takéto eseje sú veľmi dôležité najmä pri zaúčaní nových ľudí do tímu alebo k projektu.

Záver

Z tejto práce jasne vyplýva, že dokumentácia prekonala za posledné roky viaceré problémy. Napriek tomu sa však časom dostavili aj úspechy a rozšírila sa po kvalitatívnej aj kvantitatívnej stránke. Dostáva zaslúžené miesto v životnom cykle softvéru. Situácia však stále nie je ideálna. Potreba dokumentácie z hľadiska úspešnosti softvérového produktu viedla v niektorých prípadoch k jej úmyselnému kvantifikovaniu. Cieľom sa stalo dodávať klientovi hromady papiera ako dôkaz postupu práce na projekte. Toto postupne viedlo k zahlteniu klienta, čo sa veľmi negatívne odrazilo aj na klientovej ochote riešiť problémy súvisiace s projektom, prípadne na zadaní inej zákazky v budúcnosti. Softvérová spoločnosť tým dosiahla pravý opak toho, v čo dúfala.

Napriek tomu je dokumentácia softvéru potrebná, nesmie sa však zneužívať. Je to prostriedok k väčšej kvalite produktu, neslúži na zabavenie klienta. Aj tu platí, že priveľa dobrého škodí. V tejto eseji boli rozpracované viaceré možnosti ako dokumentáciu zefektívniť a ako dosiahnuť najlepší pomer medzi dĺžkou dokumentácie a jej informačnou hodnotou. Niektoré sa vyvinuli náhodou, iné kvôli potrebe zmeny.

Je potrebné dôkladne premyslieť, koľko a hlavne akej dokumentácie bude potrebné a čo z nej môže eventuálne zaujímať klienta. Preto je nutné pri jej tvorbe využívať viaceré pravidlá, aby sa nehromadila. Neúmerne veľké množstvo dokumentácie značí, že niečo nie je v poriadku a je potrebné urýchlene situáciu riešiť. Dokumentácia by mala z hľadiska komunikácie slúžiť ako prostredník medzi spoločnosťou a klientom alebo rôznymi pracovníkmi. Z pohľadu kvality sa jedná o mimoriadne silný nástroj na zvyšovanie a udržiavanie celkovej kvality projektu. Pri monitorovaní sa čiastočne osvedčuje ako ukazovateľ celkového stavu. Rôzne aspekty manažmentu vývoja softvérového produktu jasne dokazujú, že dokumentácia tvorí jeho významnú súčasť a má veľký vplyv na daný produkt.

Použitá literatúra

1. R. John Brockmann. 1987. The Mystery of the switched titles: A Review of The Soft Side of Software, A Management Approach to Computer Documentation by Theresa Foehr and Thomas B. Cross \& Creating Effective Documentation for Computer Programs by G. Prentice Hastings and Kathryn J. King. *SIGDOC Asterisk J. Comput. Doc.* 13, 1 (March 1987), 12-14. DOI=10.1145/24677.1111676 http://doi.acm.org/10.1145/24677.1111676
2. Val Silbey. 1979. Management oriented documentation of simulation. SIGSIM Simul. Dig. 10, 3 (April 1979), 29-34. DOI=10.1145/1102802.1102805 http://doi.acm.org/10.1145/1102802.1102805
3. Remco C. de Boer and Hans van Vliet. 2009. Writing and Reading Software Documentation: How the development process may affect understanding. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering* (CHASE '09). IEEE Computer Society, Washington, DC, USA, 40-47. DOI=10.1109/CHASE.2009.5071409 http://dx.doi.org/10.1109/CHASE.2009.5071409

4. Antony Bryant. 2000. It's engineering Jim \… but not as we know it: software engineering \— solution to the software crisis, or part of the problem?. In *Proceedings of the 22nd international conference on Software engineering* (ICSE '00). ACM, New York, NY, USA, 78-87. DOI=10.1145/337180.337191 http://doi.acm.org/10.1145/337180.337191
5. A.W.Lazonder, Minimalist Computer Documentation – A Study on Constructive and Corrective Skills Development, Press: CopyPrint 2000, Enschede, ISBN 90-9007479-1 http://doc.utwente.nl/58106/1/thesis_Lazonder.pdf
6. Scott W. Ambler, Best Practises for Agile/Lean Documentation, Copyright Š 2001-2012 http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm

Annotation

The influence of the quality of the documentation on creation of software product

Essay is focused on problems grouped around documentation in general and about problems, which are connected with its creation. It brings the explanation about how documentation became such a powerful tool of many different parts of project management and its saleability to client. There is a subjective point of view on value and influence of the documentation on the project along with detailed description of these two terms. Author used his personal experiences with this problem area to bring his own opinion on the influence of the documentation on the project, its value for different parts of the team, which develops software product and the creation of the documentation of course. Besides of that, he brings a critics of the lack of or abudance of the documentation and makes up alternatives, which could solve this problem even for a price of initial issues.