KOĽKO SA VENOVAŤ TVORBE DOKUMENTÁCIE PRI AGILNOM VÝVOJI.

Dokumentáciu nenávidíme dvakrát. Keď ju píšeme a keď nám chýba.

Marek Láni

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

Abstrakt. *S problémami spojenými s nedostatočnou dokumentáciou, sa počas života stretne zrejme 99% programátorov ak nie všetci. Treba povedať, že tvorba dokumentácie nepatrí k náročným procesom vývoja softvéru, avšak tento proces sa všeobecne nestretá s obľubou. Dokonca aj moderné prístupy k vývoju softvéru venujú dôležitosti jej tvorby málo pozornosti. Takýmto prístupom je aj agilný vývoj. Ten sa riadi heslom: "Dobre pracujúci softvér nad dokumentáciu" a na otázku koľko dokumentácie je potrebné odpovedá: "Akurát". V eseji však na príkladoch z praxe ukážeme, že dokumentácia je veľmi dôležitou súčasťou projektu a, že sú prípady, kedy sa dobre pracujúci softvér bez nej nezaobíde. Rovnako sa pokúsime definovať, koľko je "akurát" dokumentácie a budeme diskutovať vzory tvorby dokumentácie, týkajúce sa agilného vývoja. Týmito vzormi sú Nepravá dokumentácia, Časové značky, Projektový slovník, Elektronické zálohovanie a Zreteľná aktivita. Ku každému z týchto vzorov uvedieme príklad kedy a za akých okolností je vytváranie daného typu dokumentácie výhodné a akým problémom nám pomôže sa vyvarovať. Rovnako poskytneme ku týmto spôsobom možné vylepšenia, prípadne načrtneme negatíva alebo pozitíva jednotlivých riešení.*

Kľúčové slová: *dokumentácia, agilný vývoj, vzor, časové značky nepravá dokumentácia, projektový slovník, elektronické zálohovanie, zreteľná aktivita*

Úvod

(Na úvod by som rád oboznámil čitateľa so skutočnosťou, že pojem dokumentácia sa v tomto dokumente bude vzťahovať na internú dokumentáciu spojenú s vývojom softvéru, ktorá je využívaná čisto pre potreby tímu, pracujúceho na vývoji softvéru.)

Ako dlho písať dokumentáciu? Koľko a ktoré diagramy by boli vhodné? Presná odpoveď neexistuje. Ale to, že sme dokumentácii nevenovali dostatok času, zväčša skôr či neskôr zistíme. Pri tejto aktivite určite platí heslo: "Radšej viacej než je potrebné, ako málo". Aj keď som len študent a moje skúsenosti s programátorským svetom nie sú až také veľké, dovolím si tvrdiť, že s problémami spojenými s nedostatočnou dokumentáciou, sa počas života stretne zrejme 99% programátorov ak nie všetci. Treba povedať, že tvorba dokumentácie nepatrí k náročným procesom vývoja softvéru, avšak tento proces sa všeobecne nestretá s obľubou. Mnohým programátorom je táto aktivita proti srsti a snažia sa jej vyhnúť ako to len ide.

V dnešnej dobe moderných prístupov k vývoju softvéru, ktoré sa od dokumentácie ešte väčšmi odkláňajú, je tento problém nepostačujúcej dokumentácie badateľný ešte častejšie. Typickým prístupom, ktorý tomuto problému čelí, je agilný vývoj softvéru zameraný na požiadavky zákazníka, ktorý vyžaduje to, aby bol produkt funkčný a väčšinou o technickej dokumentácií nemá ani poňatie. Už v samotnom manifeste agilného vývoja softvéru sa píše: " Dobre pracujúci softvér nad zrozumiteľnú dokumentáciu". Ale je to skutočne tak? Dá sa vyrobiť kvalitný softvér bez dokumentácie? Určite dá, ale iba za určitých okolností. Časté tímové stretnutia pri agilnom prístupe tomu bezpochybne nahrávajú do kariet, avšak v prípade dlhotrvajúceho projektu s variabilným tímom narážame na problémy. Práve kvôli tomu sa tvorbe dokumentácie vyhnúť nedá. Prieskum v rámci agilných vývojárskych tímov [3] ukazuje, že členovia tímov zúčastnených prieskumu strávia písaním dokumentácie priemerne 10 až 20 minút denne. Väčšina opýtaných považuje dokumentáciu za dôležitú až veľmi dôležitú a značné percento označilo, že dokumentácie ku projektu je málo. Aj z týchto faktov možno usudzovať, že dokumentácii sa nevenuje dostatočná pozornosť, no treba si uvedomiť, že tvoriť a udržiavať dokumentáciu k projektu je dôležité a v určitých prípadoch jej kvalita môže mať dopad na spokojnosť zákazníka, ktorá je alfou a omegou agilného prístupu k vývoju softvéru.

Čo je to akurát?

Vráťme sa k úvodnej otázke, koľko dokumentácie stačí? Odpoveď ľudí praktizujúcich agilný vývoj je jednoduchá: "Akurát". Čiže ani málo, ale ani zbytočne veľa. No takáto odpoveď toho veľa nenapovie. Preto som sa zamyslel a odpoveďou by podľa môjho názoru mohlo byť, že je potrebné toľko dokumentácie, aby sa nevyskytla udalosť, pri ktorej nám dokumentácia bude chýbať. Aby sme takýmto situáciám predchádzali, pokúsime sa ich identifikovať a navrhnúť spôsob ich riešenia.

Zanedbaná dokumentácia

Často sa stáva, že si nás neduh vo forme odignorovania dokumentácie počká v budúcnosti, keď sa k projektu vrátime po dlhšej dobe. Avšak ešte častejšie na takéto

správanie doplácajú druhí. Konkrétne ľudia, ktorý sa do projektu zapoja počas vývoja, takpovediac za behu. Takýto nový člen tímu sa musí s projektom oboznámiť a ako náhle je dokumentácia zanedbaná a nepostačujúca, predlžuje sa čas, za ktorý sa dá do projektu dostať. Rovnako je nutné vo väčšej miere komunikovať s členmi tímu, ktorí sú s projektom dobre oboznámení a tak dochádza k plytvaniu ich časom, ktorý by mohol byť využitý efektívnejšie, nakoľko človeko-hodiny strávené objasňovaním, by sa v inom prípade dali využiť napríklad na ďalší vývoj softvéru. Niekto si ale povie, že sa to vyrovná času, ktorý by sa inak strávil písaním dokumentácie, no nie je to úplne tak. Treba počítať s faktom, že keď píšeme dokumentáciu počas tvorby softvéru, máme všetky potrebné poznatky a myšlienky čerstvo uložené v hlave a nemusíme si ťažkopádne na niečo spomínať. Taktiež sa môže stať, že sa počas tvorby softvéru priberie do tímu postupne viac ako jedna osoba a v takom prípade sa čas strávený objasňovaním projektu násobne predlžuje, pričom dokumentáciu stačí napísať iba raz. Súvisiacim problémom môže byť aj spolupráca dvoch rôznych tímov na jednom veľkom projekte, kedy tímy nezávisle od seba pracujú na dvoch susediacich záhradkách a na dokončenie projektu je nutné odstrániť plot medzi týmito záhradkami a záhradky spojiť. Bez dokumentácie je tento akt veľmi náročný a zdĺhavý a myslím si, že ani netreba vysvetľovať prečo. Problém nastáva aj v prípade, že písanie dokumentácie nie je súčasťou vývoja softvéru, ale je narýchlo písaná pri ukončovaní projektu. V tomto prípade sa dostávame k ťažkostiam opísaným vyššie a rovnako je takáto tvorba ľahšie náchylná na vznik logických chýb.

Nepravá dokumentácia

Problémy opísané v predošlom odseku sa navzájom veľmi podobajú a mohli by sme ich všetky spojiť do jednej skupiny problémov, v ktorej ide o oboznamovanie sa s projektom. Tieto problémy sa netýkajú iba agilného vývoja a možno ich nájsť aj pri iných prístupoch, no my sa zameriame ako ich riešiť v rámci agilného prístupu, s ohľadom na to, aby dokumentácie bolo "akurát". Možným riešením je tzv. **Nepravá dokumentácia** (z anglického Fake Documentation), ktorú *Hoda a kolektív* vo svojej práci [2] opisujú . Nepravá dokumentácia, je okresaná dokumentácia, ktorá je ale postačujúca na to, aby dokázala dostatočne kvalitne opísať a oboznámiť s projektom nových členov tímu, prípadne iné tímy.

Podľa rád a skúseností ľudí, ktorí majú za sebou nejaké agilne vyvíjané projekty, je dobré písať túto dokumentáciu v rámci každého šprintu, nakoľko počas šprintu sa vývojári často venujú jednej logicky ucelenej časti. Určite ste si však všimli, že sme opäť prišli k abstraktnému vyjadreniu množstva dokumentácie. Koľko dokumentácie a aká dokumentácia je potrebná na to, aby bola postačujúca? V tomto prípade zrejme pôjde najmä o funkcionálnu špecifikáciu jednotlivých častí kódu. A ako zistiť, či je postačujúca? Jednoznačne treba do procesu zahrnúť človeka, ktorý danú časť nevyvíjal a ktorý by sa po prečítaní dokumentácie vyjadril, či dokumentu rozumie a je mu z neho všetko jasné, tak povediac, dá spätnú väzbu. Jedným spôsobom je nájsť na to osobu mimo tímu, ktorá by mala na starosti práve túto aktivitu. Tu je ale potreba ďalšej pracovnej sily, ktorá by mala na starosti čítanie pomerne veľkého množstva dokumentácie. Preto si osobne myslím, že lepším riešením, by mohlo byť vzájomné vymenenie si dokumentácie medzi členmi tímu. Takýmto spôsobom by každý člen tímu mal za úlohu prečítať nanajvýš jeden dokument a

keďže na šprinty by mal byť stanovený maximálne 30 dňový interval, predpokladané množstvo dokumentácie by nemalo byť veľké. Spätnú väzbu by bolo možné vymeniť si počas scrum-u, čím by mal vo veci jasno aj scrum master. Myslím si, že tento spôsob je efektívny a časovo nenáročný, nakoľko netreba prídavnú pracovnú silu a má výhodu aj v tom, že toto riešenie napomáha a zlepšuje informovanosť všetkých členov tímu so všetkými súčasťami projektu, čo určite nie je na škodu.

Zaznamenávanie komunikácie

Ďalší problém je už špecifickejší a charakteristický najmä pre agilný vývoj. Takýto prístup je založený na zapájaní používateľa do vývoja, čo znamená, že aj počas vývoja, môže zákazník iniciovať zmeny. Po dohodnutí zmeny je potrebné prepísať resp. prekresliť kartičku s konkrétnym používateľským príbehom, ktorého sa zmena týka. Pri častých zmenách sa môže stať, že zákazník zabudne, že inicioval zmenu, alebo my si ju zabudneme zaznačiť. Takéto situácie môžu viesť až k hádkam so zákazníkom, čo je hrubé porušenie zásad agilného vývoja. Aby sme sa vyhli problémom, vhodným riešením, ako niektorí experti v agilnom vývoji radia [2], je využiť **časové značky** vykonaných zmien s pridaním informácie, kto zmenu inicioval a toto všetko elektronicky uchovať. Myslím si, že toto riešenie by sa dalo elektronickým ukladaním kartičiek s používateľskými príbehmi. Pretože pri častých zmenách môže často nastať situácia, že sa chceme vrátiť k predošlej verzii lebo sme prišli na to, že naše predošlé úsudky neboli správne. Ide o akési verziovanie týchto kartičiek, pričom podľa môjho názoru je oveľa pohodlnejšie mať verzie kartičiek uložené v počítači ako v zásuvke pracovného stola.

Nezrovnalosti v komunikácii však môžu vznikať aj medzi jednotlivými členmi tímu. Mnohí ľudia praktizujúci agilný vývoj tvrdia, že v prípade lokalitne distribuovaných tímov komunikujúcich za využitia multimediálnych nástrojov je výhodnejšie použiť textovú komunikáciu namiesto video-hovoru [2]. Prečo? Odpoveďou je ukladanie komunikácie. Môže sa zdať, že pomocou video-hovoru vieme vyriešiť potrebné veci oveľa rýchlejšie a vlastne je to tak, ale pravý benefit využitia textovej komunikácie prichádza neskôr. Prichádza práve pri riešení nezrovnalostí. V prípade využitia video-hovoru, by sme zrejme museli iniciovať ďalší hovor, no v prípade textovej komunikácie sa odpoveď nachádza v logoch a je veľmi ľahko vyhladateľná. Samozrejme takýto spôsob komunikácie nemá svoju opodstatnenosť pri malých, rýchlo hotových projektoch, no v rozsiahlejších projektoch vie textová komunikácia ušetriť čas aj keď sa to na prvý pohľad nemusí zdať. Som toho názoru, že automatické zálohovanie textovej komunikácie pri distribuovaných tímoch mierne zmenšuje znevýhodnenie takéhoto spôsobu tímovej práce, pretože v prípade lokálneho tímu sa síce komunikácia deje rýchlejšie, no počas stretnutí je dobrým zvykom udržiavať zápisnicu, ktorá zachytáva dôležité body, ktoré sa prejednávali a to je aktivita naviac.

Projektový slovník, aby sme si rozumeli

Ďalším priestorom na vznik nezhody medzi členmi tímu a zákazníkom je podľa Hodu a kolektívu [2] používanie "rôznych" jazykov. Zákazník zväčša používa biznis jazyk a programátori jazyk technický. V rámci komunikácie preto nastávajú situácie, kedy si

navzájom so zákazníkom nebudeme rozumieť, či už on nám ale my jemu. Ako to vyriešiť? Z pohľadu zákazníka jednoducho, on sa nás spýta na vysvetlenie a keďže sa v tejto situácií riadime heslom: "Náš zákazník náš pán" s radosťou mu pojem, ktorému nerozumie vysvetlíme aj dvadsať krát. Problém však nastáva v opačnom garde. Zákazníka sa nechceme pýtať viac krát na jeden pojem, jedno slovo, ktorému z jeho biznis jazyka nerozumieme. Mohlo by sa totižto stať, že sme sa ho to pýtali nie len my, ale ďalších desať členov tímu a v takom prípade, by mal plné právo byť podráždený. Takže jedna možnosť je vysvetliť si pojem po svojom, čo však v prípade zlého vysvetlenia, môže mať veľmi negatívny dopad na výsledok, ktorý zákazníkovi odovzdáme. Ako riešenie uvádza lepším spôsobom je preto zaviesť **projektový slovník**, ku ktorému majú prístup členovia tímu ale i zákazník. Takýmto spôsobom sa dá jednoducho zlepšiť komunikácia so zákazníkom, čo má priamy dopad aj na uľahčenie transformácie používateľských príbehov z biznis jazyka do technického jazyka programátora.

Zálohovanie

Jedným zo základných kameňov agilného vývoja sú fyzické artefakty. A najmä kartičky s používateľskými príbehmi, ktoré sú neraz polepené po stenách kancelárie, prípadne po celom stole. Pri väčšom počte týchto kartičiek nie je zložité stratiť organizovanosť a kartičky. Dokonca Hoda a kolektív [2] vo svojej práci predkladajú pre nezainteresovanú osobu vtipný príbeh o tom, ako upratovacia služba cez víkend spravila "jarné" upratovanie, počas ktorého sa jej zamestnancom zrejme načmárané kartičky nezdali byť dôležité a za pár minút sa im takmer podarilo pochovať projekt. Aj keď toto je extrém, ktorého pravdepodobnosť nastatia limituje k počtu jedna, no jedna dve kartičky sa utrúsia veľmi jednoducho. Riešením je už spomenuté **elektronické zálohovanie** týchto zdrojov. Ako riešenie straty kartičiek sa už v spomenutej práci uvádza tvorba fotografií. Mne sa však takéto riešenie nezdá byť tým najvhodnejším. Opäť najmä v prípade rozsiahlych projektov, by sa fotografovanie mohlo stať jednak dátovo náročnejšie, ale hlavne okrem zálohy neposkytuje žiadnu pridanú hodnotu. Namiesto fotografovania, by som osobne preferoval zapisovanie a zakresľovanie kartičiek za pomoci editovacieho nástroja, čo je možno časovo náročnejšie, ale v tomto prípade môžeme hovoriť o ľahšej znovu-použiteľnosti či už pri zmenách kartičiek alebo prípadne pri rôznych projektoch, ktoré obsahujú podobné komponenty.

Šprintuj a dokumentuj!

Viac všeobecným vzorom, ktorý sa týka agilného vývoja softvéru je vzor Zreteľná aktivita (z anglického"Distinct activity") [1], podľa ktorého, by mal byť proces dokumentácie považovaný za samostatnú aktivitu v rámci projektu, ktorá ma pridelený vlastný časový plán, prioritu a rozpočet. Osobne súhlasím s prvým bodom a myslím si, že v rámci plánovania každého šprintu, by sa mal brať do úvahy aj čas na tvorbu dokumentácie. Avšak druhý a tretí bod, pridelenie priority resp. rozpočtu pre tvorbu dokumentácie, sa už podľa môjho názoru silno odkláňajú od orientácie na zákazníka. Viem si predstaviť, že sa rozumnými argumentmi dá zákazníkovi vysvetliť, prečo je v časovom pláne zahrnutá aktivita tvorby dokumentácie. Avšak vôbec si neviem predstaviť, ako by sme sa so

zákazníkom dohadovali na rozpočte, s tým, že by v ňom bola vyčlenená položka tvorba dokumentácie. Rovnako by sme mu asi nevysvetlili, že tvorbe dokumentácie chceme prideliť vyššiu prioritu ako zapracovaniu funkcionality, ktorú on požaduje.

Záver

Aj keď je písanie dokumentácie nie veľmi obľúbená aktivita, pre vytvorenie kvalitného výsledku je nevyhnutná. A aj napriek tomu, že moderný prístup k vývoju softvéru, akým je agilný vývoj, hlása: "Pracujúci softvér nad dokumentáciu" [4], na príkladoch sme jednoznačne ukázali, že nie vždy to môže fungovať. Je však pravda, že sa niekedy môže stať, že sa dokumentácie vytvára zbytočne veľa a práve preto sme uviedli a diskutovali spôsoby a vzory, ktoré sa tomu snažia predchádzať a ukazujú ako vytvoriť dokumentácie akurát.

Použitá literatúra

1. Ruping, A.: Agile Documentation: *A Pattern Guide to Producing Lightweight Documents for Software Projects* , John Wiley, 2003.
2. Hoda, R., Noble, J., Marshall, S.: How much is just enough?: some documentation patterns on Agile projects. In: - *EuroPLoP '10 Proceedings of the 15th European Conference on Pattern Languages of Programs*, No. 5, ACM, (2010),
3. Stettina, C. J., Heijstek, W.: Necessary and Neglected? An Empirical Study of Internal Documentation in Agile Software Development Teams. In: *SIGDOC '11 Proceedings of the 29th ACM international conference on Design of communication*, ACM, (2011), 159–166.
4. Janus, A.: Towards a common agile software development model (ASDM). In: *ACM SIGSOFT Software Engineering Notes Vol.37(4)*, ACM, (2012), 1-8.

Annotation

How much time should be devoted to the creation of documentation in agile development?

The problems associated with insufficient documentation have to be faced probably by 99% of programmers if not by all. The creation of documentation is not a difficult process of software development, but this process is generally not met with much likes. Even modern approaches to software development pay limited attention to it. Such an approach is also an agile development. Its motto says: "Working software over documentation" and its answer to the question of how much documentation is necessary is: "Just enough." However, examples from practice show that documentation is a very important part of a software project and there are instances where a working software can not exist without it. We also try to define how much documentation is "just enough" and we discuss the patterns of documentation related to agile development. These patterns are Fake documentation, Timestamps, Project dictionary, Electronic backup and Distinct activity. For each of these models there are provided examples under what circumstances is creating of a specific type of documentation favorable, and what problems will it help us to avoid. We also discuss the possible ways in which it could be improved and outline the pros and cons of each solution.