AGILNE A KVALITNE

Kvalita je to, keď sa vracia zákazník a nie tovar.

Maroš Unčík

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

Abstrakt. *Slovo kvalita sa s vývojom softvérových produktov spája už od počiatkov vývoja softvéru. Vieme, že kvalita má niekoľko definícií, no z pohľadu každej je kvalita vo vývoji softvéru nevyhnutná. V súčasnej dobe sa presadzujú agilné metódy pri vývoji softvéru. Agilné metódy sa používajú najmä v malých, či stredne veľkých tímoch a umožňujú vyvíjať softvér pomerne rýchlo vzhľadom na nestabilné požiadavky. Otvorenou otázkou ale je, či sú agilné metódy vhodné pre projekty každej veľkosti a zároveň, ktoré časti v manažmente kvality sú kritické. Ako teda vôbec v projektoch môžeme zabezpečiť kvalitu pri agilnom vývoji softvéru? Na čo si v malom tíme dávať pozor a aké úskalia z pohľadu kvality prinášajú? Je zabezpečenie kvality pri agilnom vývoji softvéru závislé od prostredia? Hoci na tieto otázky nie je v literatúre vyhradený názor, je potrebné sa nad nimi zamyslieť, pretože ich ignorovanie môže mať negatívny vplyv na úspech projektu.*

Kľúčové slová: *zabezpečenie kvality, agilný vývoj, malý tím, malá organizácia, veľká organizácia*

Úvod

Už na začiatku vývoja softvéru sa prišlo na to, že chaotický vývoj softvéru je zlý. Výsledkom chaosu vo vývoji softvéru je nesprávny produkt, ktorého dodávka je často oneskorená a nastáva následné predraženie produktu. Toto všetko sa podpisuje na kvalite výsledného produktu, čo má negatívny vplyv na istotu zákazníka. Ako vhodný liek na tento problém sa ukázalo zavedenie riadeného procesu pri vývoji softvéru.

Myslím si, že zavedenie procesu vo vývoji softvéru prinieslo predvídateľnosť a zamedzenie opakovaniu chýb, hoci na úkor spomalenia samotného vývoja a zníženia reakcie na zmeny. Jedným z prvých procesov, prebraný pôvodne z priemyslu, ktorý sa

začal vo vývoji softvéru používať bol vodopádový model. Jeho typický znakom je, že do ďalšej etapy sa prechádza až po úplnom ukončení aktuálnej etapy. Nedodržaním tohto pravidla by malo za následok nutnosť vrátiť sa do predchádzajúcej fázy a tým zvyšovanie nákladov. Vodopádový model sa dnes ešte stále uplatňuje vo vývoji softvéru a rovnako aj v mnohých oblastiach v priemysle.

Hoci sa vodopádový model používa od 70. rokov a je jedným z najstarších modelov, ukázalo sa, že vodopádový vývoj softvéru má mnoho nedostatkov. Podľa môjho názoru je najväčším nedostatkom najmä rôzna miera abstrakcie v jednotlivých etapách vývoja. To robí vodopádový model často nepoužiteľným pre menšie softvérové projekty. Softvér je produkt na jednom z najvyšších stupňov abstrakcie a je nevyhnutné, aby jeho vývoj prebiehal vo viacerých iteračných cykloch.

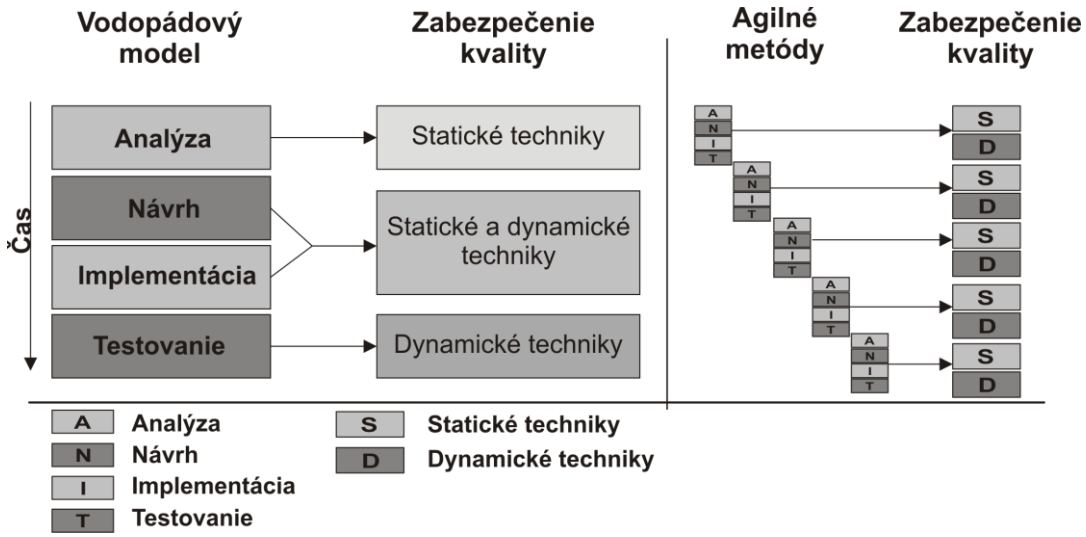
Súčasne s tým ako sa softvérové systémy stávali rozsiahlejšími a komplexnejšími, bolo potrebné neustále reagovať na zmeny. Softvér sa predsa musí prispôsobovať používateľovi a nie naopak. Preto sa vodopádový model v praxi nahradil modelmi, ktoré sa od tohto modelu odvíjajú – snažia sa odstrániť nevýhody vodopádového modelu, ale často reflektujú aspoň kúsok vodopádového modelu. Len nedávno nastúpili na rad agilné metódy. Tieto šikovné metódy schopné ľahko a rýchlo riešiť zadané problémy umožňujú vytvoriť produkt v rekordne rýchlom čase. Sú navrhnuté tak, aby boli vhodné skôr pre menšie tímy. V praxi sa často rozdelia veľké tímy na podtímy, ktoré riešia určitú časť problému [4].

Medzi výhody agilných metód patrí snaha akceptovať individuálny prístup a kreativitu programátorov a s výhodou ich používať. Hoci toto je podľa môjho názoru práve jednou z najdôležitejších motivácií a dôvodom rýchleho presadzovania agilných metód, otázkou je, či je to naozaj výhoda.

Nezávisle od výberu metódy pri tvorbe softvéru je nevyhnutnou podmienkou úspechu projektu zabezpečenie kvality. Ak sa na kvalitu pozrieme ako na dva základné procesy – zabezpečenie celkovej kvality – čiže zapracovanie samotnej túžby na kvalitný produkt počas vývoja a proces verifikácie a validácie [2], musím skonštatovať, že zabezpečenie kvality v agilných metódach oproti zabezpečeniu kvality v tradičnom vodopádovom vývoji softvéru je veľmi rôzne. Zatiaľ čo vo vodopádovom modeli na začiatku prevládajú statické techniky zabezpečovania kvality (najmä skúmanie a kontrola dokumentácie a zdrojových kódov), v strednej fáze statické a dynamické a nakoniec dynamické techniky. Pri agilných metódach sa vo všetkých fázach využívajú aj statické aj dynamické techniky zabezpečovania kvality (pozri Obr. 1). Dynamické techniky sa využívajú vo všeobecnosti viac a zároveň oveľa častejšie v priebehu vývoja. Pri zrýchlenom tempe vývoja sa tak v agilných metódach potláčajú niektoré nástroje v procese zabezpečenie kvality. Ak sú tieto nástroje v tradičných spôsoboch vývoja takmer nevyhnutné, ako je teda možné, že nám tieto techniky v agilnom vývoji nechýbajú? Dokážeme teda aj pri tak rýchlom vývoji softvéru zabezpečiť potrebnú kvalitu? Je kvalita softvérových produktov vôbec zaručená? Do akej miery?

Kvalita je kľúčom k úspechu

Čo je to vlastne kvalita softvérového produktu? Jeho to jeho rýchle dodanie? Robustnosť? Nízka cena? Všetky tieto veci dohromady? Alebo sme azda na niečo zabudli?



Obr. 1. Techniky zabezpečenia kvality [2].

Môžeme kvalitu uchopiť?

Kvalitu ako takú nie je jednoduché slovne vyjadriť. Určite sa však rozhodne zhodneme na tom, že to je vlastnosť. Túto vlastnosť sa svojimi definíciami snažili ohraničiť mnohí [3], no mojim názorom je, že sa to ani jednému autorovi nepodarilo vyjadriť. Inak by sme predsa nepolemizovali o jej význame.

Myslím si, že kvalita je pre každého subjektívny pojem. Ak by sme sa z pohľadu trvácnosti produktu pozreli na ozdobnú gýčovú sošku, mohli by sme prehlásiť, že je to kvalitný produkt. Ak sa však na túto sošku pozrieme z pohľadu jej praktickosti, zhodneme sa na tom, že soška je v tomto ohľade veľmi nekvalitná. Soška nemá žiadnu funkciu, ktorá by napríklad uľahčovala život. Zvyčajne je len položená na poličke a padá na ňu prach. Napriek tomu si však takú sošku kúpi množstvo zákazníkov.

Ako sa však soška dostane k zákazníkovi? Ako vlastne výrobca sošiek prinúti zákazníka, aby si vybral práve jeho výrobok z množstva ďalších? Odpoveďou je, že na úspechu danej sošky sa podieľa množstvo ďalších faktorov, či už vo výrobe alebo pri konečnom zhodnotení produktu. Tie majú dopad na celkový úspech produktu. Kvalita tak nie je závislá od jednej vlastnosti produktu, ale od celej rady vlastností, ktoré do veľkej miery dokáže výrobca ovplyvniť.

Ak sa bavíme o kvalite softvéru, musím konštatovať, že jej ponímanie sa v čase mení. Nedávna doba vyvolala túto zmenu v požiadavkách na kvalitu v softvérových produktoch. Pred mnohými rokmi sa softvér prispôsoboval hardvéru, keďže cena softvéru bola ďaleko nižšia ako cena hardvéru. Preto sa v tomto období kvalita zameriavala viac na efektívne využitie hardvéru. V dnešnej dobe však softvér cenovo prekonal hardvér, a preto sa kvalita zamerala viac na softvérový produkt ako taký, najmä na proces jeho tvorby.

Druhým aspektom je uspokojenie potrieb zákazníka. Preto sa často tvrdí, že kvalita je miera splnenia zákazníkových požiadaviek. S týmto tvrdením musím súhlasiť.

Softvér sa v dnešnej dobe zapája do ľudského života veľmi vysokou mierou. Zapájajú sa i do takých činnosti ako je lekárstvo, energetika a finančníctvo, kde softvérový produkt musí poskytovať maximálnu spoľahlivosť. Preto má kvalita v softvérových produktoch svoje dôležité miesto.

Malý tím vs. veľký projekt

„Budeme agilní - urobíme tak veci lepšie." Myslím, že podobné myšlienky prúdia v hlavách tých, ktorí práve prechádzajú na agilný vývoj softvéru. Nie je sa čomu diviť. Agilné metódy prežívajú svoj veľký rozmach, pričom marketing vytvára agilným metódam dobé meno. Toto meno je čiastočne aj zaslúžené, pretože agilné praktiky vo vývoji softvéru prinášajú na prvý pohľad pomerne bohaté možnosti zabezpečovania kvality počas tvorby produktu. Tieto techniky sa zakladajú najmä na neustálej komunikácií a konzultáciách so zákazníkom, presadzujú sa dynamické techniky, ako sú časté prototypovanie a testovanie, či revízia kódu (pozri tabuľku Tab. 1). Tieto, na prvý pohľad bohaté možnosti zabezpečovania kvality, majú však dopad na ďalšie vykonávané činností.

Tab. 1. Niektoré agilné techniky zabezpečovania kvality používané v jednotlivých fázach vývoja.

Technika	Fáza vo vývoji softvéru
Vytvorenie metafory systému	Definovanie požiadaviek a návrh systému a softvéru
Vytvorenie prototypu	
Komunikácia so zákazníkom	
Čiastkové testovanie	Implementácia a testovanie častí
Revízia kódu	
Programovanie v páre	
Priebežné nasadzovanie	Integrácia systému a jeho testovanie
Akceptačné testy	
Spätná väzba od zákazníka	

Techniky zabezpečovania kvality v agilných metódach boli, podľa môjho názoru, navrhnuté s ohľadom na ich jednoduchosť a flexibilitu použitia. Cieľ bol, aby ich mohli využívať najmä malé tímy, pracujúce v úzkom spojení. Preto sa vo väčšine prípadov jedná o dynamické metódy, ktoré sa zakladajú často na automatizovaných postupoch, na vykonávaní kódu a jeho predvádzanie. Statické metódy ako kontrola dokumentácie alebo statické prezeranie kódu sa dostali do úzadia. Hoci sa tieto pôvodné techniky takmer nevyužívajú, myslím si, že nám v konečnom dôsledku pri agilných metódach nechýbajú. V agilných metódach máme za ne náhrady. Na druhej strane problémy, ktoré by inak vznikli napríklad pri vodopádovom vývoji softvéru, v agilným metódach preto nepociťujeme resp. nepociťujeme ich vždy. Priblížme si to na dvoch prípadoch.

Predstavme si projekt menšieho rozsahu, ktorý sa môže podobať na školské projekty, aké sú často riešené na univerzitách. Tím, ktorý rieši takéto úlohy sa typicky skladá

z menej ako 8 členov a rieši úlohu, ktorá často nie je zo vzdialenej doménovej oblasti [1]. Nemusí tak investovať do zdĺhavých analýz, ktoré by zaberali podstatnú časť z celkového času projektu. To znamená, že tím nevytvorí veľké množstvo dokumentácie a zároveň vo všeobecnosti platí, že takýto malý tím je s dokumentáciou oboznámený na veľmi dobrej úrovni. Na strane druhej je v malom tíme každý člen tímu pravidelne informovaný o splnených úlohách a celkom dianí v tíme. Vzniká tak akýsi globálny dohľad nad zabezpečením kvality, za ktorú zodpovedá každý člen tímu. Pokroky a prípadné zmeny sú často konzultované navzájom, a preto je možnosť omylu v prípade nedodržania požiadaviek značne menšia.

Ako som naznačil menšie množstvo dokumentácie tak vzniká najmä z dôvodu, že členovia tímu sa sústredia na vytváranie samotného produktu. Problémovú oblasť poznajú, a preto často osobne nepociťujú potrebu vytvárať dokumentáciu. Myslím si však, že menšie množstvo dokumentácie môže spôsobiť problém, ktorý sa pri tradičných spôsoboch vývoja nemusia prejaviť. Týmto problémom je, že samotný proces je vďaka nedostatku dokumentácie menej sledovateľný, čo môže znížiť samotnú kvalitu. Často tak nedokážeme identifikovať a analyzovať problémy, ktoré sa vyskytli a zabrániť ich opakovaniu. Vhodné riešenie takéhoto problému vidím najmä v nepodcenení dokumentácie a jej starostlivej kontrole. Aj dobre vypracovaná dokumentácia sa podieľa na celkovej kvalite produktu.

Situácia je iná, ak máme projekt veľkého rozsahu. Takýto projekt môže prebiehať niekoľko rokov a je doň zapojených niekoľko desiatok ľudí, odborníkov z celého sveta. Predstavme si, že cieľom projektu je vytvorenie softvéru pre včasné varovanie krajín pred tsunami. Ak by sme takýto projekt chceli riešiť nasadením agilnej metódy, nevyhli by sme sa ani v tomto prípade rozsiahlej a časovo náročnej analýze, pri ktorej by sme vytvorili značné množstvo dokumentov. Nie je možné, aby sa v týchto kvantách dokumentov dokonale vyznal každý člen tímu. Z toho dôvodu si myslím, že je potrebné zabezpečiť určitú kvalitu týchto dokumentov, a preto, hoci to nie je štandardné, aj agilný tím siahne po statických metódach zabezpečovanie kvality – akým je napríklad recenzia dokumentu. Viac sa tu vynára otázka či je tento projekt vôbec vhodný na riešenie formou agilných metód a či nie je vhodnejšie siahnuť po inej metóde vývoja.

Prejsť, či neprejsť – začať, či nezačať?

Veľký rozdiel v nasadzovaní agilných metód si všimneme, ak existujúca organizácia migruje zo zaužívaných techník a postupov. Formujú sa tak nové agilné tímy zo starých, ktoré musia riešiť iné problémy, ako keď vzniká úplne nový tím. Malé spoločnosti takýto prechod zvládajú ľahšie. Ak si zoberieme príklad veľkej spoločnosti, ktorá využíva dlhoročné postupy založené na vodopádovom modeli, nie je pre ňu ľahké a ani možné skokovo prejsť na agilný vývoj softvéru.

Veľké spoločnosti riešia často veľmi rozsiahle projekty, ktoré prebiehajú po dlhé časové obdobie. Predstavme si v takejto spoločnosti veľký tím, ktorý obsahuje desiatky ľudí a rieši dlhoročný projekt. Snaha prejsť na agilné postupy im tak pri existujúcom projekte môže spôsobiť nemalé problémy najmä pri zabezpečení [4]:

- regresného testovania veľkého projektu,
- zmeny stratégie testovania z manuálnej na automatickú,

- vykonávania testovania po častiach,
- pružného a rýchleho testovania v krátkych časových intervaloch.

Zvládnutie týchto problémov nie je triviálne a ich nedostatočné zvládnutie má za následok zníženie kvality výsledného produktu. Podobné problémy začínajúce tímy nemajú, pretože ich vyriešenie je prirodzenou súčasťou vývoja. To im prácu značne uľahčuje. Myslím si, že práve malé tímy, ideálne práve vzniknuté, sa dokážu agilným postupom a metódam prispôsobiť oveľa jednoduchšie.

Problémom však je, že takéto tímy ešte nemajú dostatok skúseností a môžu určité aspekty podceniť. Napríklad statická kontrola zdrojových kódov, ktorá často v agilnom procese vývoja softvéru chýba, môže byť prospešná. V končenom dôsledku sa odrazí na celkovej kvalite produktu. Statickú kontrolu však nie je nutné vykonávať vždy. Napríklad pri programovaní v páre sa explicitne nevykonáva statická kontrola kódu, no dvojica členov tímu pri programovaní je taká zohratá, že prípadne chyby v programe sa ich spoločným úsilím značne minimalizujú.

Keďže agilné metódy sledujú viac individuálnych členov tímu a snažia sa akceptovať rozdiely medzi ľuďmi, posúvame sa k vývoju softvéru, ktorý je viac riadený ľuďmi ako samotným procesom. Takáto dogma automaticky zamieta fakt, že ľudia sú nahraditeľnými komponentmi v tíme a samotný tím a aj celý proces vývoja softvéru je tak do značnej miery závislý od jednotlivcov. To samozrejme ovplyvňuje aj kvalitu. Strata jedného z členov tímu, z rôznych dôvodov (nespokojnosť s platovými podmienkami, nezhody a pod.) tak môže mať veľký dopad na celkovú kvalitu projektu. Týmto neduhom trpia najmä menšie spoločnosti, ktorých rodinné prostredie trpí odchodom jedného z členov tímu omnoho viac ako vo veľkej spoločnosti. Veľké firmy sa s úbytkom pracovníkov vysporiadajú oveľa ľahšie a podľa môjho názoru tak vytvárajú v tomto ohľade oveľa kvalitnejšie a stabilnejšie podmienky.

Agilné metódy vývoja nie sú vhodné pre každú spoločnosť. Rozhodnutie či prejsť na agilný vývoj, prípadne či začať s agilným vývojom softvéru, treba zvážiť vzhľadom na povahu projektu, veľkosť spoločnosti a typ projektu. Agilné metódy sa vidia použiteľnejšie na menšie projekty, ktoré prebiehajú menšie časové obdobie, no nemusí to byť vždy pravidlom. Kvalita v agilných tímoch sa potom od týchto vlastností odvádza. Je teda nielen závislá od veľkosti firmy, v ktorom sa softvérový produkt vyvíja, ale viaže sa aj na samotné prostredie firmy, rovnako aj na doménu softvérového produktu.

Záver

Je viacero kľúčov k úspešnosti projektu. Niektoré kľúče sú menšie a iné zas väčšie. Som presvedčený o tom, že zabezpečenie kvality je jeden z najväčších kľúčov k otvoreniu úspešného softvérového produktu. V súčasnosti nie je pravidlo, že najväčšie firmy sú aj najúspešnejšie. Aj malé neznáme spoločnosti, ktoré zamestnávajú len zopár zamestnancov, či zanietení jednotlivci sa zo dňa na deň môžu stať úspešnými. Je ale dôležité si uvedomiť, že svet sa skladá z detailov a na každej drobnosti, ktorá môže rozhodnúť o našom úspechu, záleží. Preto treba byť pripravený a nezanedbať niektorú časť v kvalite produktu.

Ako som sa snažil poukázať, v súčasnosti presadzované agilné metódy nie sú zárukou úspechu a globálne neodstraňujú všetky problémy. Je tomu práve naopak a

generujú dokonca niektoré ďalšie problémy. Aj z agilnej metódy tak môže vzísť nekvalitný produkt, ktorému na kvalite nepridáva ani fakt, že bol vytvorený na základe najnovšej šikovnej techniky.

Sám som priaznivcom agilných metód a názory, ktoré som vyvodil, hovoria skôr v prospech agilných metód. Je potrebné brať do úvahy aj prostredie a doménu ich použitia. Závery, ktoré som uviedol, sú tak pre mňa ponaučením a nútia ma zamyslieť sa nad tým, aký bude vývoj agilných metód a aké trendy vývoja softvéru prídu po nich.

Použitá literatúra

1. Marrington, A., Hogan, J.M., Thomas, R.: Quality assurance in a student-based agile software engineering process. In: *Proc. of the 2005 Australian Conference on Software Engineering*, IEEE Computer Society, Washington, DC (2005), 324-331.
2. Ming, H., Verner, J., Liming Z., Babar, M.A.: Software quality and agile methods. In: *Proc.s of the 28th Annual international Computer Software and Applications Conference*, IEEE Computer Society, Washington, DC (2004), 520-525.
3. Mnkandla, E., Dwolatzky, B.: Defining Agile Software Quality Assurance. In: *Proc. of the international Conference on Software Engineering Advances*, IEEE Computer Society, Washington, DC (2006), 36-42.
4. Shaye, S.D.: Transitioning a Team to Agile Test Methods, In*: Proc. of the AGILE '08 Conference*, IEEE Computer Society, Washington, DC (2008), 470-477.

Annotation

Agility and Quality

The word quality is bound with the software development from its beginning. We know that quality has several definitions, but independently from it, the quality of project is necessary. At present, agile methods are promoted in software development. These methods are mainly used in small or medium-sized software developing teams and allow relatively quickly handle unstable requirements. The question that still remains open is, if the agile methods are suitable for projects of all sizes and which parts of management of quality are critical in a project. How to ensure the quality in agile software development? What pitfalls should be a small team aware in terms of quality? Is quality assurance in agile software development environmentally depended? Although these questions are not restricted in the literature view, it is necessary to think about them. Ignoring these important questions could have a negative impact on a project success.