VPLYV AGILNÝCH METÓD NA KVALITU SOFTVÉRU

Dosahovaný výsledok sa odvíja od zvoleného postupu.

Marek Sobôtka

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

Abstrakt. *Cieľom každého zákazníka je dostať softvér v požadovanej kvalite, ktorú bude akceptovať. Samotný softvér ale môžeme vyvíjať viacerými spôsobmi. Či chceme alebo nie, zvolená metóda vývoja sa odrazí na celkovej kvalite konečného produktu. V súčasnosti sa čoraz častejšie začínajú používať agilné metódy vývoja, ktoré sa rozširujú na úkor klasických prístupov. Vďaka nim produkujeme softvér rýchlejšie, ale je nutné vedieť zabezpečiť kvalitu na požadovanej úrovni v každej etape. Na základe porovnania agilného prístupu s klasickým vývojom ukážem, ako závisí kvalita softvéru od vybraného spôsobu vývoja a čím je ovplyvnená. Aké dôsledky z toho pre nás vyplývajú? Ako ich využiť v náš prospech a maximálnu spokojnosť zákazníka?*

Kľúčové slová: *agilný vývoj, testovanie, kvalita, vodopádový model*

Úvod

V súčasnosti prenikajú počítače do každej oblasti nášho bežného života. Čoraz viac ľudí si svoju bežnú prácu bez pomoci týchto elektronických pomocníkov nevie predstaviť. Vďaka nim dokážeme urýchliť a zautomatizovať často opakované činnosti a minimalizovať počet chýb, ktoré vznikajú. Avšak srdcom každého počítača je softvér. Inak by to bola len kopa elektroniky bez svojho opodstatnenia. Pred tým ako softvér začneme používať, musí ho niekto vytvoriť. Samotná tvorba softvéru je veľmi rozsiahla a náročná činnosť, ale cieľ je vždy rovnaký. Programy sa vytvárajú za určitým účelom tak, aby splnili, čo sa od nich očakáva v požadovanej kvalite.

Kvalita

Čo však rozumieť pod samotným pojmom kvalita? Táto pomerne jednoduchá otázka môže byť interpretovaná viacerými spôsobmi a záleží len na uhlu pohľadu, z ktorého ju sledujeme. Podľa normy ISO 8402 je kvalita definovaná ako *„súhrn vlastností a charakteristík entity, ktoré preukazujú jej schopnosť uspokojiť určené alebo odvodené potreby“* [2]. Pod entitou v našom prípade môžeme rozumieť samotný softvér. Na každý softvér sú kladené rôzne požiadavky a hlavne v rozdielnom rozsahu, v závislosti od cieľovej skupiny. Úplne inak budeme vnímať kvalitu pri programe na spracovanie štatistických informácií, kde v najhoršom prípade dostaneme nepresné výsledky a inak pri programe na obsluhu jadrového reaktora, kde sebe menšia chyba môže spôsobiť katastrofu nemalého rozsahu. Jednotlivé vlastnosti a charakteristiky kvality softvéru opisuje norma ISO 9126, ktorá ich rozdeľuje do šiestich hlavných kategórií [2]. Tie sú ďalej rozdelené na podkategórie, tak ako je uvedené v Tab. 1.

Tab. 1. Charakteristiky kvality softvéru podľa ISO 9126 [2].

Charakteristika	Príslušné podkategórie
Funkcionalita	Presnosť, vhodnosť, interoperabilita, vyhovenie, bezpečnosť
Spoľahlivosť	Zrelosť, tolerancia k chybám, obnoviteľnosť, dostupnosť
Použiteľnosť	Pochopiteľnosť, naučiteľnosť, prevádzkyschopnosť
Efektívnosť	Správanie sa v čase, využitie zdrojov
Udržovateľnosť	Analyzovateľnosť, možnosť zmeny, stabilita, testovateľnosť
Prenositeľnosť	Prispôsobivosť, inštalovateľnosť, schopnosť koexistencie, zhodnosť, vymeniteľnosť

Pri kvalite softvéru si všímame tri aspekty. Jedná sa o čas, náklady a splnenie požiadaviek. Tieto aspekty sú vzájomne závislé. Zníženie času, ktorý je k dispozícii na dokončenie programu, sa negatívne prejaví zvýšením nákladov a možným nesplnením požiadaviek. Toto je jeden z dôvodov, prečo je pri tvorbe softvéru dôležité správne si vybrať metódu vývoja.

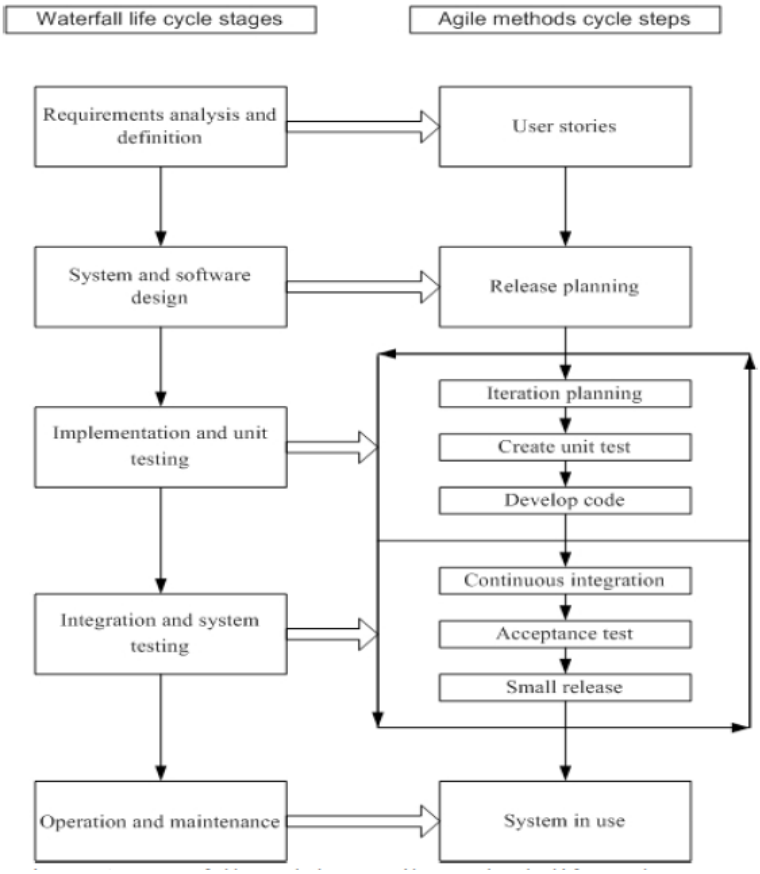
Vodopádový model

Od konca šesťdesiatych rokov bolo vyvinutých niekoľko metodológií, ako vyvíjať softvér. Tie sa postupom času zlepšovali a zdokonaľovali. Medzi tradičné spôsoby vývoja patrí vodopádový model, ktorý je najstarším procesným modelom vývoja softvéru. Počas svojej existencie bol nespočetne krát využitý pri vývoji či už malých alebo veľkých projektov. Hoci je už pomerne starý, stále nachádza svoje využitie aj dnes. Tento model spočíva v tom, že životný cyklus vývoja softvéru rozdeľuje do piatich lineárnych a nezávislých fáz. Nasledujúca fáza nemôže začať, pokiaľ predchádzajúca neskončila. Výsledok z nej sa potom použije na začiatku nasledujúcej fázy. Najprv sa zanalyzujú požiadavky, ktoré sú na plánovaný systém kladené. Následne sa na základe tejto analýzy vytvorí návrh systému. Ďalšou fázou je samotná implementácia, v ktorej sa systém vytvorí a otestuje. Potom prebehne integrácia a systémové testovanie. V poslednom kroku sa systém nasadí do reálnej prevádzky a prebieha jeho údržba. [3]

Táto metóda má však viacero nedostatkov. Pre zákazníka je často problém sformulovať všetky svoje požiadavky hneď na začiatku. Niektoré z nich sú nejasné a môžu sa objaviť rozpory. Vďaka tomu, že jednotlivé fázy vývoja sú od seba nezávislé, prispôsobenie sa novým požiadavkám je nesmierne náročné, ba niekedy aj absolútne nemožné. Problémy spôsobuje aj fakt, že zákazník vidí systém až na konci procesu a môže sa stať, že vyvinutý systém nenapĺňa zákazníkove predstavy.

Agilný vývoj

Spomínané problémy sa snažia riešiť agilné metódy vývoja. Popularitu začali získavať koncom deväťdesiatych rokov. Tieto metódy sú založené na viacerých technikách. Medzi najdôležitejšie z nich patrí jednoduché plánovanie, kratšie iterácie, skoršie vydanie projektu a častejšia spolupráca so zákazníkom. Vďaka nim je možné dodať produkt za kratší čas ako pri vodopádovom modeli. Tieto metódy boli navrhnuté aj za účelom lepšieho odolania neustálym zmenám zákazníkových požiadaviek a meniacemu sa prostrediu. Na rozdiel od tradičného prístupu, agilný vývoj zmeny víta. V podstate ide o vodopádový model, ktorý sa opakovane vykonáva v kratších časových obdobiach. Názorne je to zobrazené na Obr. 1. V každom kroku sa pokračuje vo vývoji z predchádzajúcej iterácie a dosiahnutý výsledok sa konzultuje so zákazníkom. [3]



Obr. 1. Životný cyklus vodopádového modelu a agilnej metódy [3].

Ako ale tieto rozdiely medzi metódami vývoja ovplyvnia kvalitu celkového produktu? Môžeme sa na tento problém pozerať z hľadiska, ako dobre je vytvorený systém otestovaný. Predpokladá sa, že čím viac testov sa vykoná, tým viac chýb sa odhalí a odladí a tým viac je výsledný produkt kvalitnejší. Všeobecne rozlišujeme dve skupiny techník na zabezpečenie kvality – statické a dynamické [3]. Statické techniky sa nevykonávajú nad kódom projektu, ale sú napríklad založené na štúdiu dokumentácie k systému. Odhalenie chyby v dokumentácii znamená odhalenie chyby v projekte. Dynamické techniky naopak pracujú s kódom, môžu to byť automatizované testy alebo akceptačné testy, či sa systém chová podľa očakávania.

Tradičný verzus agilný vývoj

Problémom zabezpečenia kvality sa zaoberali na technickej univerzite v Lappeenranta vo Fínsku. Vypracovali štúdiu v ktorej sa zamerali na rozdiely v postupe testovania. Do štúdie bolo zahrnutých dvanásť firiem rozdielnych veľkostí. Niektoré z nich vyvíjajú tradičným spôsobom, zvyšné agilne. Dáta, ktoré zozbierali, získali pomocou rozhovorov so zamestnancami na rôznych pozíciách. Rozhovory boli nahrávané pre následné spracovanie. Rozhovor sa viedol s návrhárom resp. programátorom, manažérom vývoja resp. testovania a s testerom resp. programátorom zodpovedným za testovanie. Po spracovaní údajov dospeli k piatim hypotézam. [1]

Hypotézy

„Agilné praktiky majú tendenciu nechať viac času na testovanie, pričom celkové množstvo času na projekt zostáva rovnaké.“ [1]

S touto hypotézou sa plne stotožňujem. Vďaka tomu, že agilné metódy vyvíjajú v kratších cykloch, v rámci ktorých musia byť všetky činnosti úplne hotové, kým sa bude pokračovať v ďalšej iterácii, je nutné mať každú takto dokončenú časť projektu správne otestovanú. Tým pádom prebieha testovanie častejšie a v menšom rozsahu [3], ale celkové množstvo času na testovanie je väčšie, ako keď sa testovaniu pridelí samostatná fáza. Dôvodom, prečo si to myslím je, že testovanie sa často podceňuje, robí sa na poslednú chvíľu pred odovzdaním projektu. Ľudia zodpovední za testovanie sú tak pod väčším tlakom, vykonávajú testy narýchlo a ľahko sa tak niečo prehliadne.

Pre agilnú metódu hrá aj fakt, že testovanie sa vykonáva v čase, keď programátori ešte majú v pamäti kód, ktorý naprogramovali, na rozdiel od klasického prístupu. Tým pádom majú lepší prehľad v tom, čo testujú. Nemusia si spomínať, čo ktorý kód vlastne vykonáva. Takto ušetrený čas môžu využiť na samotné testovanie.

Myslím si však, že to nie je vždy také ideálne. Môže nastať situácia, keď sa konkrétna malá iterácia na projekte nestíha a potom sa testovanie priveľmi urýchľuje, tak ako sa to môže stať v prípade vodopádového modelu. Toto riziko je však pri agilnej metóde menšie a ak aj takáto situácia nastane, vždy môže byť dobehnutá v nasledujúcej iterácii projektu.

„Uplatnenie agilných praktík zjemňuje vyťaženie testovacích zdrojov, avšak nezníži ich celkové množstvo počas celého projektu.“ [1]

Túto hypotézu si vykladám tak, že testerov nepotrebujeme nárazovo, keď prebieha fáza testovania u klasického prístupu, ale že sú vyťažení priebežne, v každej iterácii agilného vývoja. Dochádza tak k rozloženiu zdrojov na celú etapu vypracovávania projektu. Vďaka tomu je možné lepšie prideľovať zdroje, keďže vyťaženie ľudí je priebežné. Testovanie je možné rozdeliť tak, aby nebolo nutné prideľovať na viacero projektov jedného človeka. Už z predchádzajúcej hypotézy to má pozitívny efekt, že tester má lepší prehľad, čo testuje a vďaka tomu sa môže do testu viac zahĺbiť. Jednotlivé testy na seba nadväzujú, preto výsledky testov z predchádzajúcej iterácie vývoja je jednoducho možné využiť a pokračovať ďalej.

Čo sa týka celkového času, tak tu by som si dovolil tvrdiť, že množstvo testovacích zdrojov sa nielen nezníži, ale dokonca sa zvýši. Vďaka čomu je to možné? Tým, že testovanie prebieha koordinovane v každej iterácii, tak ako už bolo spomínané, zo samotnej podstaty agilného vývoja vyplýva, že cieľom je zvýšiť množstvo času venovaného testovaniu. Ruka v ruke s tým idú aj zdroje, keď v každom okamihu musí test niekto vykonávať. Čiže čím viac času sa strávi na testovaní, tým viac zdrojov je nutné na to vynaložiť.

„V záujme efektívneho vývoja a následného testovania je nutné, aby zainteresovaní rozumeli a prispôsobili sa princípom agilného vývoja.“ [1]

Táto myšlienka je úplne prirodzená. Darmo sa vymyslí nejaká super metóda ako postupovať, keď sa podľa nej nikto nebude riadiť a nikto jej neporozumie. Už som naznačil, že agilné metódy nie sú vhodné pre každý tím. Pre efektívne fungovanie je nevyhnutné, aby bol tým zladený a správne fungovala koordinácia medzi členmi tímu. Ak toto nie je dosiahnuté, jednotlivé úlohy sa nestíhajú včas, všetko sa posúva alebo sa znižuje čas strávený nad niektorými úlohami, ako je práve testovanie. Dôsledkom toho je negatívny dopad na celkovú kvalitu, pretože je to v kontradikcii s pôvodným predpokladom, ktorý vraví, že čím viac testujeme, tým kvalitnejší produkt dostaneme. U tímu, ktorý si navzájom rozumie je väčší predpoklad, že sa prispôsobia princípom, ktoré sú na nich kladené agilnou metódou.

Taktiež je nevyhnutné zvýšiť komunikáciu a spoluprácu so zákazníkom, pretože iba on sám najlepšie vie, čo požaduje. Každá zainteresovaná osoba musí požiadavkám rozumieť, aj keď nie sú jasne špecifikované od začiatku, ale postupne ich zákazník dopĺňa. Postupné zjemňovanie špecifikácie docieli lepšie naplnenie požiadaviek. U tradičného vodopádového modelu je táto možnosť vylúčená, pretože nato proste nezostáva v samotnej fáze testovania dostatok času. Viacnásobné spresnenie požiadaviek za tak krátky čas je nemožné splniť. Rozbitie produktu na menšie časti u agilnej metódy teda zvyšuje efektívnosť testovania a tým aj vývoja. Rizikom je, ak zákazník nemá záujem podieľať sa na vývoji. Tento stav môže nastať, keď očakáva zadaný výstup a nemá čas konzultovať zmeny. V takomto prípade je akýkoľvek agilný postup kontraproduktívny, pretože by prišiel o jeden z hlavných pilierov samotnej metódy.

„Interný zákazník podporuje zavádzanie agilných metód.“ [1]

Áno, určite na tom niečo je, pretože zákazník chce vidieť do procesu. Ako prebieha, ako sa postupuje, čoho sa dosiahlo. Nehrá pri tom rolu, či je interný alebo externý. K tomuto sa dostane iba pri agilnom vývoji, preto ak bude schopný ovplyvniť, ako sa bude vyvíjať, tak pokiaľ je rozumný, bude presadzovať agilný postup. Pri tradičnom vývoji totiž vidí zákazník výsledky až na konci procesu vývoja. Predpokladá sa, že interný zákazník a externý zákazník sa dobre poznajú a budú presadzovať rovnaké záujmy, čo zvýši harmóniu pri vývoji. Taktiež programátori pri akejkoľvek nejasnosti v špecifikácii získajú informácie, ako to bolo v skutočnosti myslené oveľa skôr, ako keby si museli u vedúceho tímu vybaviť, aby im to zistil.

Avšak si myslím, že interný zákazník je reálny iba pri tímoch s väčším počtom ľudí, inak by bolo málo aktívnych členov priamo zapojených do programovania a realizácie projektu. V menšom tíme podľa mňa úplne postačuje komunikácia s jedným zákazníkom, aby neprichádzalo k zahlteniu tímu priveľa informáciami. Z vlastnej skúsenosti na projekte menšieho rozsahu viem, že sa potom môže ľahko stať, že požiadavky sa budú spresňovať takým štýlom, že sa bude práca hromadiť, namiesto toho aby postupom času ubúdala. Potom je nutné odkladať dátum dokončenia projektu a to je nepríjemné ako pre zákazníka tak aj pre tím samotný.

„Uplatnenie agilnej metódy umožňuje rýchlejšiu reakčnú dobu na zmenu a manažment funkcií pri testovaní.“ [1]

Myslím si, že táto myšlienka je tiež úplne jasná. Je to jedna zo základných myšlienok agilného vývoja. Zmena je vítanou súčasťou projektu. Vývojom v menších iteráciách ľahšie naplánujeme a zapracujeme zmeny, či už v aktuálne prebiehajúcej iterácii alebo v priebehu nasledujúcich. Pri vodopádovom modeli je možné zmeny zapracovať až po dokončení projektu.

Agilný vývoj zvyšuje kvalitu

Na základe naštudovaných informácií si dovolím vysloviť vlastnú hypotézu, že agilná metóda vývoja zvyšuje kvalitu softvéru. Kvalitu ovplyvňujú tri aspekty – čas, náklady a požiadavky. Tie sú od seba závislé a teda, keď sa nám podarí venovať viac času na testovanie vďaka agilnej metóde, pozitívne sa to odrazí na lepšom splnení stanovených požiadaviek. Získavame väčšiu istotu, že funkcie systému budú správne fungovať, takže vlastne zvýšime kvalitu systému samotného. Zvýšime tým spokojnosť zákazníka, ktorý tak získa funkčný, dostatočne otestovaný systém, ktorý bude spĺňať jeho požiadavky vo väčšej miere, ako by tomu bolo u tradičného vývoja. A to všetko najmä vďaka lepšiemu rozloženiu času, keď na testovanie zostáva viac času. Je možné, že to v druhom rade môže znížiť aj náklady, ktoré by sa museli vynaložiť na údržbu systému, keďže je predpoklad, že systém vyvíjaný agilne bude mať odhalených a opravených viac chýb, ktoré sa v priebehu vývoja objavili.

Moja hypotéza však s veľkou pravdepodobnosťou nebude fungovať v prípade, že tím ľudí, ktorý ide vyvíjať agilnou metódou, nie je zohraný. Je tu väčšie riziko, že tím nebude efektívne komunikovať, čo sa môže odraziť na neskoršom dokončení pridelených úloh.

Ako už bolo spomínané skôr, menej času je nutné niekde kompenzovať a často to skončí práve u testovania. S menším časom stráveným testovaním správania a funkcionality systému sa tím pripraví o jednu z výhod, ktorú mu agilný vývoj prináša. S takýmto prístupom sa kvalita znižuje na úroveň tradičného vývoja alebo ešte nižšie, v závislosti od toho, či sa zameškané testy vykonajú v ďalšej iterácii vývoja systému.

Záver

Aké dôsledky z toho teda pre nás vyplývajú? Dosahovaný výsledok sa odvíja od zvoleného postupu. Preto by mal tím vyvíjať agilne, pokiaľ to povoľuje povaha projektu, tím je dostatočne zohraný a nemá strach skúsiť novú metódu. Všetko závisí od toho ako zodpovedne sa potom k agilnej metóde vývoja tím postaví a ako dôsledne bude dodržiavať jeho zásady a využívať možnosti, ktoré mu poskytuje. Odmenou je získaná vyššia kvalita, ktorá môže zvýšiť reputáciu tímu alebo firmy ale hlavne zvýšenie spokojnosti zákazníka.

Použitá literatúra

1. Kettunen, V., Kasurinen, J., Taipale, O., Smolander, K.: A study on agility and testing processes in software organizations. In: *Proceedings of the 19th international Symposium on Software Testing and Analysis*, Trento (2010), 231-240.
2. Bevan, N., Azuma, M.: Quality in use: incorporating human factors into the software engineering lifecycle. In: *Software Engineering Standards Symposium and Forum, 1997. 'Emerging International Standards'. ISESS 97., Third IEEE International*, Walnut Creek (1997), 169-179.
3. Huo, M., Verner, J., Zhu, L., Barbar, M.A.: Software quality and agile methods. In: *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, Hong Kong (2004), 520-525.

Annotation

Impact of agile methods for software quality

The aim of every customer is to get software at the required quality, he/she will accept. The software itself can be developed in several ways. Like it or not, the chosen method of development is reflected in the overall quality of the finished product. Currently, the usage of agile development methods raise and are expanding at the expense of traditional approaches. Thanks to them we produce software faster, but we need to know how to ensure the required quality level in each stage. I will show how the software quality depends on the chosen method of development, based on a comparison of agile development approach with a classical development, and what affects it. What are the consequences of this result for us? How to use them in our benefit and maximum customer satisfaction?