Agilné metódy vývoja softvéru a rozsah projektu

TOMÁŠ KLEMPA

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
klempa05@student.fiit.stuba.sk

Abstrakt. Tradičné metódy vývoja softvéru už prestávajú byť postačujúce, v súvislosti s požiadavkou na rýchly vývoj softvéru a jeho časté zmeny. Zákazníci požadujú dodávku softvéru v čo možno najkratšom čase a v čo možno najvyššej kvalite (a primeranej cene). Čas a kvalita sú premenné projektu, ktoré sú ovplyvnené rozsahom projektu. Rozsah projektu je spočiatku pevne stanovený, ale po istom čase sa začne sa začne meniť, čo môže ohroziť úspech projektu. So zmenou rozsahu však treba vopred počítať. Podľa zdrojov použitých v tejto eseji vyplýva, že riadenie rozsahu projektu možno efektívnejšie docieliť s použitím agilného vývoja. Esej sa zaoberá otázkou čo nám môžu agilné metodológie ponúknuť, definuje rozsah a s nim súvisiace základné premenné projektu, naznačuje spôsob ako možno efektívnejšie riadiť rozsah projektu, vysvetľuje vplyv neistoty a zložitosti v projekte. V závere je spomenutých niekoľko tipov, ktoré by mohli napomôcť pri agilnom vývoji v súvislosti s rozsahom projektu.

Úvod

Známy fakt, že softvér sa mení vplyvom okolia je skutočne pravdivý. Zákazník chce dostať za svoje peniaze vysokú kvalitu, a keď ju aj dostane po istom čase (alebo aj ihneď) by chcel mať vo svojom softvéri toto a toto, napr. kvôli potrebám spoločnosti. Všeobecne povedané ide o vylepšenie, alebo zmenu softvéru. Zákazník chce vymeniť zastaraný systém, ktorý už nevyhovuje jeho potrebám. Softvér sa musí prispôsobiť keďže zákazník žiada zmeny v softvéri.

Nasleduje identifikácia nových čŕt, ktoré by mal „vylepšený“ systém obsahovať. Dátumy vývoja boli stanovené na základe najlepších odhadov času potrebného pre vývoj týchto čŕt. Nasleduje podrobný návrh týchto čŕt, čo je veľmi dôležité. V prípade, že bude nová črta implementovaná nekorektne (rozmaznaný zákazník si to predstavoval ináč), spôsobí to oneskorenie odovzdania systému. Celý projekt môže byť

Manažment v softvérovom inžinierstve, december 2005, s. 1-9.

vážne ohrozený. Kvalita produktu je veľmi dôležitá a pre dodávateľa snáď nejestvuje nič horšie ako strata dôvery zákazníka.

Oneskorený projekt bude dodaný načas, ale vývojový tím bude pracovať nadčasy, čo sa odrazí na ich produktivite. Reťazová reakcia určite platí aj v softvérom inžinierstve a teda zníženou produktivitou utrpí aj kvalita projektu, v dôsledku čoho môže nastať prípad kedy zákazník „objaví“ chyby zmeneného systému, ktorý tak túžobne očakával. Pravdepodobne ešte horší prípad môže nastať, keď sa dodanie systému oneskorí (aj o niekoľko mesiacov!). To už zákazník pení a v jeho očiach dodávateľ stráca kredit, čo môže vyústiť do stavu, kedy sa zákazník v budúcnosti bude rozhodovať pre systém konkurenčného dodávateľa.

Iba prednedávnom sa objavili na „scéne“ agilné metódy vývoja softvéru. Nie sú však všeliekom na neduhy softvérového inžinierstva ani sa o to (našťastie) nesnažia. Poďme sa pozrieť bližšie na agilné metódy a čím môžu byť užitočné pre nás.

Čo nám môžu ponúknuť agilné metodológie

Dovolím si stručne ozrejmiť pojem agilné metodológie, aby v ďalšom texte nedošlo k nejednoznačnosti. Agilné metodológie označuje skupinu metodík, ktoré vychádzajú z poznania, že jedinou cestou ako overiť správnosť navrhnutého systému, je čo najrýchlejšie ho vyvinúť, predložiť zákazníkovi a na základe spätnej väzby upravovať[4].

Agilné metodológie sľubujú zlepšenie výkonu projektu. Neznie to príliš nadnesene? Sľuby v softvérovom inžinierstve vyznievajú príliš nedôverčivo, ale pokúsme sa pozrieť bližšie.

Praktiky ako *testom riadený vývoj*, *jednoduchý návrh* a *refaktoring* môžu zvýšiť kvalitu kódu.

Krátke iterácie nám ponúkajú viac informácii o tom ako sme na tom s projektom a poskytujú rýchlu spätnú väzbu od zákazníka.

Scrum praktiky definujú denné stretnutia a pravidelné demonštrácie na konci mesiaca. Zapájajú aj zákazníka minimálne raz do mesiaca.

Informačné radiátory (radiators) predstavujú udržiavanie informácií o *interakčnom (interactive) návrhu*, objektových modelov a iteračných plánov. Informácie sú umiestnené fyzicky na stene kde umožňujú kontinuálne informovať tím.

Niekoľko atribútov *extrémneho programovania (XP)*: programovanie v pároch (zvýšenie kvality kódu), krátke iterácie a neustále testovanie.

Pre detailnejšie popisy jednotlivých praktík odporúčam čitateľovi naštudovať si príslušnú literatúru, napr. aj [1].

Druhá strana mince alebo štatistiky neklamú

Zaujala ma štatistika [3] o znalostiach projektových manažérov zo spoločností, ktoré používajú tradičné metodiky vývoja (plan-based). Oslovených bolo 20 manažérov (10 tradičných a 10 agilných spoločností), pričom ich priemerný vek je 40 rokov. 14

spoločností sídlia v Taliansku, 5 v USA a 1 v Švajčiarsku. Štatistika hodnotila znalosti manažérov o agilných metódach, ich výhodách a nevýhodách.

Približne 90% manažérov vedia o agilných metódach, hoci ich nepoužívajú. Hlavnou príčinou pre neprijatie agilných metód sú povrchné znalosti o nich, odpor v spoločnosti a zákazníkov a veľké geograficky oddelené tímy. Odpor zo strany zákazníkov sa mi pozdáva nemiestny – od kedy určujú zákazníci aké metódy má dodávateľ používať? Zrejme netušia, že agilné metódy kladú dôraz na vývoj softvéru spolu so zákazníkom.

Ďalšia otázka sa týkala znalostí o XP a metóde Scrum. Všetky spoločnosti, ktoré používajú tradičné aj agilné metódy vývoja softvéru poznajú XP, ale iba 60% agilných spoločností a žiadne z „tradičných“ spoločností poznajú metódu Scrum.

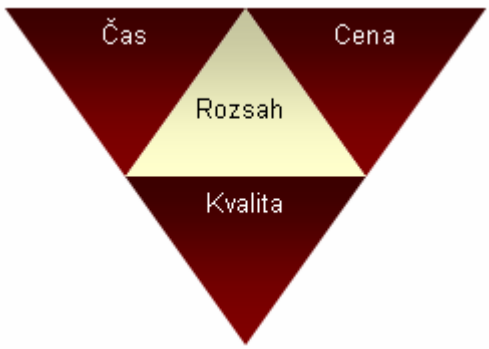
Čísla pôsobia až šokujúco, autori v článku tvrdia, že výsledky tejto štatistiky nereprezentujú celkovú populáciu v softvérových spoločnostiach.

Štyri premenné

Vychádzajme z faktu, že existujú štyri základné premenné vývoja softvéru (podľa [1]):

1. cena,
2. kvalita,
3. čas a
4. rozsah.

Čas a cena sú v podstate výstupom rozsahu. Ak máme definovaný rozsah, môžeme vypočítať čas (ako dlho nám to bude trvať) a cenu (koľko nás to bude stáť). Nesprávne definovaný rozsah negatívne ovplyvní čas a cenu.



Obr. 1.Štyri premenné projektu

Premenné je vhodné v skratke opísať a vysvetliť ako vzájomne pôsobia.

Cena

Túto premennú možno ovplyvniť zmenou ľubovoľnej inej premennej. Veľký vplyv na cenu majú ľudia participujúci na projekte. Je zrejmé, že pridaním dvojnásobku počtu ľudí sa vývoj projektu nezrýchli, naopak, je pravdepodobné, že sa spomalí vzhľadom na problémy komunikácie v rámci tímu, keďže dvojnásobná bude komunikácia v tíme.

V poriadku, pridajme teda do projektu niekoľko ľudí (menej ako dvojnásobok) a uvidíme čo sa bude diať. Pri stavbe domu by sa práca pravdepodobne urýchlila, ale to pri vývoji softvéru v tejto situácii opäť neplatí. Efekt by sa dostavil až po dlhšom čase. Podstata tkvie v tom, že novému členu tímu chvíľu trvá, než spozná zabehnutý systém vývojového tímu. Hoci urobí určitú časť práce za iných členov tímu, členovia tímu mu musia isté veci vysvetliť a tým ich akoby okrádal o čas. Dovolím si citovať Fredericka P. Brooksa: „Pridaním ďalších ľudí do projektu, ktorý mešká, ho urobí ešte viac meškajúcim.“[2].

Kvalita

Podľa [1] možno rozlíšiť kvalitu na:

- externú – viditeľná zákazníkom (napr. nefunkčné požiadavky) a
- internú – určuje „vnútorné” kvality systému: dobrý návrh, testovanie atp. Vnútorná kvalita je veľmi dôležitá a má vplyv na rýchlosť vývoja projektu (chabá vnútorná kvalita má za následok spomalenie). Z tohoto dôvodu napr. kladie XP dôraz na testovanie a refaktoring.

Čas a rozsah

Sú premenné najvhodnejšie pre riadenie, ktoré sú spočiatku fixné, ale neskôr sa musia nevyhnutne meniť a narastať (napr. nové podrobnosti, ktoré nie sú na začiatku predvídateľné), čím môže utrpieť kvalita. A následkom toho, že čas a rozsah sa mení, vieme presne povedať, kde sme až na konci projektu.

Z tohto dôvodu sa javí ako výhoda projekt „ukončovať” v pravidelných intervaloch (rádovo v týždňoch) – iteráciách. Iterácie nám pomáhajú sledovať účinky meniaceho sa rozsahu projektu, ktoré môžu byť fatálne pri tradičných metódach vývoja softvéru (pri použití vodopádového modelu).

Outsourcing a plánovanie v XP

Outsourcing je v súčasnosti veľmi frekventovaný pojem. Akým spôsobom však ovplyvňuje rozsah projektu? Odpoveď nájdeme v outsourcingovej zmluve. Podľa [1] obvyklá outsourcingová zmluva stanovuje tri zo štyroch základných premenných: rozsah, čas a cenu. Tradičné biznis vzťahy potrebujú miernu úpravu, pokiaľ chceme plánovať a vykonať projekt s XP.

Termín “premenná” nie je použitý náhodou. Ak sa objaví nečakaná situácia, jedna z premenných sa zmení (napr. čas). Od kvality, ktorá je defacto najťažšie zmerateľná

premenná, nečakané situácie majú tendenciu ovplyvniť práve ju. Prakticky to znamená menej testovania, menej návrhu, a menej komunikácie v rámci tímu, alebo so zákazníkom. Akonáhle máme nezáme množstvo práce, rozsahu,času a ceny, úspech projektu je vážne ohrozený.

Autori [1] vidia základný problém v pevne stanovenej zmluve a to zmysle, že vo svojich záujmoch idú dodávateľ softvéru a zákazník navzájom proti sebe. Zákazník chce za svoje peniaze čo možno najväčší rozsah, kým dodávateľ chce urobiť čo možno najmenej práce. Tento fakt však nemusí výrazne ovplyvniť projekt, pokiaľ ide vývoj podľa plánu. Situáciu možno opísať nasledovným príkladom: zákazník podpíše zmluvu s dodávateľom a za svoje veľké peniaze, ktoré zaplatil trvá na tom, aby mu dodávateľ presne oznámil čo dostane. Má na to samozrejme právo, ale kde je potom problém? Plánovanie je nie je predvídanie budúcnosti [1]. Táto veta, myslím, hovorí za všetko. Zákazník môže sledovať vývoj v pravidelných intervaloch, napr. každé dva týždne.

Dodanie softvéru v čas a vo vysokej kvalite?

Jeff Patton v [6] sa zaoberá, akým spôsobom možno docieliť efektívnejšie riadenie rozsahu projektu. Použil agilný vývoj a interakčný (interaction) návrh.

Interakčný návrh predstavuje množinu praktík, ktorých prínosom je primerané správanie systému. Návrh možno rozdeliť na dve fázy. Prvá fáza sa zameriava na ľudí a skúma čo je ich cieľom. V ďalšej fáze, ktorá nadväzuje na získané ciele, vytvoríme najmenšiu a najjednoduchšiu množinu úloh, ktoré umožňujú ľudom používať ich softvér. Zámerom je teda splniť tieto ciele. Interakčný návrh je svojou koncepciou výrazne odlišný od tradičného návrhu.

Prínos autorovho príspevku vidím v skutočnosti, že agilné metodológie môžu za určitých podmienok zlepšiť riadenie rozsahu projektu. Pokladal som za dôležité v skratke interpretovať autorovu ideu a odporúčania vyplývajúce z tohto článku sú napísané v kapitole Recept na spokojný život alebo rady pri agilnom vývoji.

Ak čitateľa táto myšlienka zaujala, odporúčam prečítať si článok [6].

Zložitosť a neistota projektu

Zaujal ma článok [5], v ktorom autor kategorizuje projekty do štyroch skupín v súlade s ich zložitosťou a neistotou.

Autor na základe projektov riešených v minulosti zistil, že existujú dva základné atribúty ktoré ovplyvňujú typ použitého procesu: zložitosť a neistota.

Začal vyšetrovaním Crystal metódy Alistaira Cockburna a aj prístupu Barryho Boehma založeného na riziku (z dôvodu získania väčšieho množstva atribútov projektu), pre zmiešanie agílneho a plánom riadeného (tradičného) vývoja softvéru. Vymenoval všetky kritické atribúty a zistil dva základné: zložitosť a neistotu.

V nasledovných dvoch podkapitolách opíšem základnú myšlienku zložitosti projektu a detailnejšie neistotu projektu, ktorá súvisí aj s flexiblitou rozsahu projektu.

Zložitosť

Autorov tím vyvinul systém pre bodovanie zložitosti projektu na základe: veľkosti tímu, nevyhnutnosti úlohy, sídla tímu, vyzretosti tímu (výkonnosti tímu), rozdielov medzi znalosťami v oblasti a závislosti s ďalšími projektmi.

Veľkosť tímu

Veľkosť tímu ma majoritný vplyv na zložitosť projektu. Ako som uviedol v predchádzajúcom texte, pridaním dvojnásobného počtu ľudí do tímu, projekt dvojnásobne neurýchli. Často sa vedú diskusie o tom, či sú agilné metodológie vhodné pre veľký tím, to už je však otázka pre inú esej.

Dôležitosť projektu

Dôležitosť projektu má tiež majoritný vplyv na vývoj. Je zrejmé, že zložitosť projektu bude narastať spolu s jeho dôležitosťou.

Sídlo tímu

Sídlo tímu v jednej miestnosti umožňuje výbornú komunikáciu medzi členmi tímu. Naopak, tímy rozdelené vo väčších vzdialenostiach, alebo tímy sídliace vo viacerých časových pásmach zvyšujú zložitosť projektu.

Výkonnosť tímu

Veľký rozdiel medzi tímom zloženým zo skúsených vývojárov a tímom zloženého z úplných začiatočníkov.

Rozdiely medzi znalosťami v oblasti

S klesajúcimi znalosťami vývojárov v danej oblasti narastá zložitosť projektu.

Závislosti

Atribút určuje stupeň závislosti projektového tímu na ďalších skupinách alebo na iných projektoch v spoločnosti. Tesná spolupráca s ďalšími projektmi môže zvýšiť zložitosť projektu.

Neistota

Základné atribúty neistoty projektu (viď. aj obrázok č.2) sú: neistota trhu, technická (odborná) neistota, dĺžka projektu a iné projektové závislosti na danom projekte a flexibilita rozsahu projektu.

Atribúty neistoty a ich bodové ohodnotenie					
Atribút	1	3	5	7	10
Neistota trhu	Známy produkt, pravdepodobne definovaný zmluvný záväzok	Očakávaná minimálne zmeny na cieľovom trhu	Počiatočný odhad cieľového trhu pravdepodobne k požiadavke riadenia	Podstatná neistota trhu	Nový, neznámy a neodskúšaný trh
Technická neistota	Zlepšenie existujúcej architektúry	Vieme ako to vytvoriť	Nie sme si celkom istý ako to vytvoriť	Zložitý prírastkový výskum	Nová technológia, nová architektúra
Dĺžka projektu	1 - 4 týždne	6 mesiacov	12 mesiacov	18 mesiacov	24 mesiacov
Závislosti, flexibilita rozsahu	Dobre definované zmluvné záväzky	Rozsah nie je veľmi flexibilný	Rozsah má miernu flexibilitu	Rozsah je veľmi flexibilný	Veľmi vysoká závislosť

Obr. 2. Atribúty neistoty a ich bodové ohodnotenie, zdroj: [5]

Neistota trhu

Ak sú potreby trhu dobre známe, projekt nebude potrebné príliš riadiť a naopak. Riziko môže predstavovať prienik spoločnosti na nový trh.

Technická neistota

Projektové tímy vyvíjajúce nové produkty často využívajú najnovšie technológie, takže budú mať vyšší stupeň neistoty. Vyzreté produkty používajú overené technológie, a preto budú mať minimálnu technickú neistotu, hoci sa môžeme stretnúť so zvýšením technickej neistoty pri pridaní novej technológie do existujúceho produktu.

Dĺžka projektu

Čím bude dĺžka projektu vyššia, tým väčšia je šanca, že produkt ovplyvní technická alebo trhová neistota. Podľa definície rozsahu a obrázku č.1 je dĺžka projektu ovplyvnená rozsahom projektu. Navyše, ak sa dodanie produktu oneskorí, môže tým utrpieť kvalita.

Závislosti a flexibilita rozsahu

Vyšší stupeň závislosti nášho projektu na iných projektoch môže obmedziť úroveň riadenia projektu. To samozrejme súvisí s rozsahom projektu (vyššia flexibilita spôsobí vyššiu neistotu).

Recept na spokojný život alebo rady pri agilnom vývoji

Autor [6] na základe vyriešených projektov, vytvoril zoznam, ktorý sa snaží nasledovať pri vývoji nového projektu. Zoznam ma zaujal a z tohto dôvodu pokladám

za vhodné uviesť niekoľko bodov, ktoré by mohli pomôcť čitateľa nasmerovať pri agilnom vývoji softvéru. Obsahy jednotlivých bodov som opísal vlastnými slovami.

- *Ponechajte návrh všeobecný a rozsah jemný.* Autor odporúča identifikovať ľudí, ktorí budú používať systém, úlohy a priority. Je známe, že pri agilných metódach sa takmer zatracuje návrh a preto autor odporúča vyvarovať sa podrobnému návrhu obrazoviek, návrhu tabuliek databáz a pod. Tieto detaily môžu rozptyľovať pri detailnej identifikácii čŕt a ich priorít.
- *Zákazník nie je protivník.* Ak sa chce zákazník podieľať a pomôcť, neexistuje dôvod prečo ho odmietnuť!
- *Vytvorte plán spolupráce.* Vytvorenie plánu spolupráce (collaboration) so zameraním na detaily. Plán by mal obsahovať činnosti, ktoré budú riešené spolu so zákazníkom napr. spoločné stretnutia. Ďalší bod, ktorý potvrdzuje môj názor, že agilné metódy sú „ľudskejšie“.
- *Fázy dodávky.* Autor odporúča rozdeliť projekt aspoň na dve fázy (v nezávislosti na veľkosti projektu). Prvú fázu dodávky by si mal zákazník možnosť vyskúšať. Úspech prvej fázy projektu zvýši dôveru zákazníka v projekte a tíme. Prvá fáza navyše umožňuje odhaliť, čo zákazník naozaj požaduje od produktu a čo môže byť v produkte vynechané. To sa odrazí vo finálnej dodávke.
- *Plánujte aj vynechanie čŕt.* Uvoľnenie každej fázy projektu by malo obsahovať aj niekoľko čŕt s nízkou prioritou. Konštrukcia softvéru by mala dovoliť jednoduché odstránenie, alebo znemožnenie nekompletných čŕt. Taktiež by mala byť dodávka softvéru možná.

Záver

Ako z textu vyplýva, agilné metodológie skutočne môžu efektívnejšie „riadiť“ rozsah projektu. Vhodnými praktikami možno predísť (alebo ich obmedziť) negatívnym následkom ako napr. oneskorenie dodania produktu, alebo pokles jeho kvality.

Počnúc od plánovania až po dodávku produktu je výhodná aktívna spolupráca so zákazníkom. Ako pozitíva by som zdôraznil hlavne iteračný vývoj a pravidelné stretnutia so zákazníkom spojené s prezentáciou už hotovej časti produktu.

Myslenie projektových manažérov neovplyvním, ale myslím si, že by nemali zatvárať oči pred novými metódami, ktoré by mali efektívnejšie pomáhať pri vývoji softvéru. Ich spoločným cieľom je predsa dodávka kvalitného produktu a dôvera zákazníka.

Použitá literatúra

1. Beck, K., Fowler, M.: *Planning Extreme Programming First Edition*. Addison-Wesley, 2000.
2. Brooks, F. P.: *The Mythical Man-Month: Essays on Software Engineering Anniversary Edition*. Addison-Wesley, 1995.
3. Ceschi, M., Sillitti, A., Succi, G., Panfilis, S. D.: Project Management in Plan-Based and Agile Companies. *IEEE Software*, Volume 22, No. 3 (2005), 21 – 27.
4. Kadlec, V.: Programujte agilně, nic jiného vám nezbýva! A nebo ano? In: http://www.zive.cz/h/Programovani/AR.asp?ARI=110219, (naposledy aktualizované 8.4.2003).
5. Little, T.: Context-Adaptive Agility: Managing Complexity and Uncertainty. *IEEE Software*,Volume 22, No. 3 (2005), 28 – 35.
6. Patton, J.:Unfixing the Fixed Scope Project: Using Agile Methodologies to Create Flexibility in Project Scope. *In: Agile Development Conference 2003*, p. 146. http://www.abstractics.com/papers/UnfixingScope.pdf
7. Turbit, N.: Defining the Scope of a Project. In: http://www.projectperfect.com.au/downloads/info_define_the_scope.pdf (naposledy aktualizované 27.6.2005)

Annotation

Agile software development methods and project scope

Traditional software development methods are not sufficient in cases of requirements on rapid software development and its repeat changes. Customers require fast delivery with high quality (and appropriate price). Time and quality are variables of project, which are influenced by scope. In the beginning, project scope is fixed, but after some time it started to change (raising), which could have negative influence on success of project. We have to know that project scope changes, in the beginning of project. References used in this paper support a fact that scope managing could be more effective using agile development. This paper deals with question: what should agile methodologies for us, defines project scope and other project variables, shows the more effective way of project scope managing, writes influnence of complexity and uncertainty to project. At the end of this paper, tips for help during agile development (in association of project scope) are written.