Agilné plánovanie

MICHAL OLÁH

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

Abstrakt. Plánovanie je jednou z najdôležitejších častí softvérového projektu. Riadenie softvérových projektov sa v poslednom čase viac a viac orientuje na nové metodiky ako agilné a extrémne programovanie. V takýchto projektoch je plánovanie čiastočne posunuté do úzadia a aj napriek tomu sú takéto projekty úspešné. Plánovanie v takýchto projektoch sa používa hlavne pri stanovovaní cieľov, pričom samotné plánovanie práce je ponechané na pracovníkov. Existuje ale nemalá časť projektov kde sa využívajú agilné metódy, v ktorých by presnejšie plánovanie pomohlo naplniť ciele projektu oveľa lepšie. Cieľom tejto eseje je poukázať na riziká pri vyžití agilných metód a ponúknuť alternatívy na zlepšenie manažmentu pomocou plánovania.

Úvod

Plánovanie patrí k najdôležitejším častiam projektov. Riadenie projektov sa v poslednom období stále viac a viac orientuje na nové metodiky, ktoré plánovanie využívajú v obmedzenej miere v klasickom zmysle slova plánovať. Hovoríme samozrejme o agilných metódach resp. tzv. agilných projektoch Pri týchto metódach sa intenzívne komunikuje so zákazníkom a ciele sa dynamicky menia podľa aktuálnej situácie. Otázkou teda zostáva, či má plánovanie v zaužívanom ponímaní daného termínu zmysel. Dovolím si tvrdiť, že nie, avšak plánovanie má rozhodne veľký význam aj pri takýchto druhoch projektov. Treba však brať do úvahy špecifiká agilných projektov a plánovať podľa toho. V rámci svojej knihy Mike Cohn pekne zhrnul dôvody čoho sa máme držať pri plánovaní v agilných projektoch [2]:

1. Plánovať treba často.
2. Odhady veľkosti a času potrebných na vykonanie úlohy sa vykonávajú oddelene.
3. Plánovanie robí na rôznych úrovniach.
4. Plány sú založené na vlastnostiach produktu a nie úlohách.
5. Malé príbehy pomáhajú s pracovným tempom.
6. Rozrobené úlohy sú eliminované s každou iteráciou.

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-8.

7. Monitorovanie sa deje na tímovej úrovni.
8. S neistotou sa počíta a plánuje sa na ňu.

Za každým bodom by sa dal nájsť jeden (prípadne viac) neúspešných projektov, kde z nejakého dôvodu zlyhalo plánovanie. V nasledujúcich častiach sa pokúsim priblížiť najpodstatnejšie body bližšie, ako aj ilustrovať ich na príkladoch.

Prečo plánovať často?

Pri plánovaní sa často stáva, že sa hneď na začiatku projektu je snaha vytvoriť dokonale presný plán. Ide samozrejme o nesplniteľnú úlohu, o ktorú sa snažia mnohí nielen pri plánovaní v klasických projektoch, ale aj v agilných projektoch. Omnoho lepší prístup k danej problematike opísal Mike Cohn [2], a tým je, že sa má plánovať často v podstate po každej iterácií.

Na to, aby bol plán použiteľný, musí byť presný, avšak musíme mať na pamäti, že každý plán vytvorený na začiatku bude nepresný. Jeden z dôvodov prečo aktualizujeme plán je, aby sme postupne odstránili nepresnosti. To znamená, že v prvotnom pláne môžeme predpokladať, že dodáme 300-400 „príbehových bodov“ v treťom štvrťroku. Tento plán sa môže ukázať ako dobrý (dodali sme 305 „príbehových bodov“ v auguste), ale tento plán nie je veľmi presný. Pre začiatok projektu môže však byť tento plán s najväčšou pravdepodobnosťou dostatočným podkladom pre rozhodovanie. Aby bol plán aj naďalej užitočný, bude sa musieť spresniť, a môže tvrdiť, že v rámci projektu dodáme 380-400 „príbehových bodov“ v septembri.

Pri agilnom odhade a plánovaní sa berie na vedomie, že naše vedomosti sú vždy nedokonalé, a vyžaduje sa, aby sa plán menil podľa toho, ako sa učíme. Ako sa tím postupne učí o produkte, ktorý vytvára nové požiadavky, sa pridávajú do celkového plánu. Ako sa tím postupne zoznamuje s technológiami, ktoré používa, a ako dobre spolu spolupracujú rýchlosť práce sa upravuje. Na to, aby plán zotrval užitočný, tieto poznatky musia byť zakomponované do plánu.

Ako na odhady?

Z vlastnej skúsenosti viem povedať, že plánovanie len na základe veľkosti je značne nepresné a neúplné. Keď sa totiž odhaduje systémom „toto je stredne ťažká obrazovka, táto zaberie štyri hodiny a táto je ľahká, tá zaberie do jednej hodiny“, môže sa stať, že na tú ťažkú sa dostane skúsenejší programátor a za tri hodiny ju naprogramuje, avšak keď ľahšiu dostane programátor s menšími skúsenosťami, môže mu zabrať rovnaký čas. Plánovanie preto musí brať v úvahu aj rýchlosť, teda množstvo abstraktných „príbehových bodov“, ktoré tím ako celok vie zvládnuť za jednu iteráciu[2].

Častou chybou v plánovaní (bez ohľadu na to, či ide o tradičné alebo agilné tímy) je spájanie odhadov veľkosti a času, za ktorý sa daná úloha dá splniť. Veľkosť a čas potrebný na vykonanie danej úlohy sú prepojené, avšak dĺžku ovplyvňuje veľa ďalších faktorov. Na projekte budú pracovať programátori s rôznymi skúsenosťami. Podobne

dĺžka je ovplyvnená veľkosťou tímu. Štvorčlennému tímu môže projekt trvať šesť mesiacov (24 človekomesiacov). Osemčlenný tím môže zredukovať tento čas na štyri kalendárne mesiace ale 32 človekomesiacov napriek tomu, že projekt je rovnaký v oboch prípadoch.

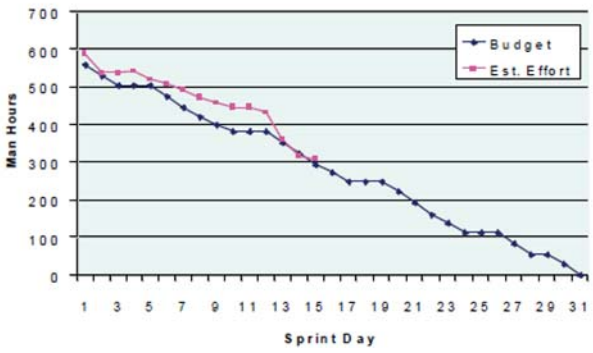
Agilné odhady a plánovanie má úspech, pretože odhaduje veľkosť a dĺžku oddelene. Začína sa odhadom veľkosti projektových „príbehov“, ktoré sa ohodnocujú „príbehovými bodmi“. Pretože tieto body sú abstraktné, sú odhadom veľkosti. Následne odhadujeme rýchlosť, a keď spojíme rýchlosť s veľkosťou, dostaneme odhad dĺžky trvania projektu. Našim odhadom a plánovaciemu procesu veľmi pomôže takéto jednoznačné oddelenie veľkosti a dĺžky trvania.

Sledovanie postupu sa vykonáva na úrovni tímu

Jedna z vecí, ktoré výrazne uľahčuje plánovanie v agilných projektoch je fakt, že sledovanie postupu sa vykonáva na úrovni tímu. Poviete si, že to určite vnáša istý druh netransparentnosti a môže spôsobiť, že niektorí pracovníci sa môžu „flákať“. Opak je však pravdou.

Tradičné prístupy k odhadovaniu a plánovaniu sledujú a odmeňujú pokrok na úrovni každého člena tímu. Toto vedie k rôznym problémom. Napríklad ak ukončenie úlohy skôr bude mať za následok, že programátor bude potrestaný, tak sa programátor jednoducho naučí pracovať tak, aby neskončil skôr. Namiesto toho aby skončil skôr a zahlásil úlohu ako splnenú, bude čakať s nahlásením až do termínu. Agilné odhady a plánovanie sa úspešne vyhýbajú takýmto druhom problémov tým, že sledujú postup iba na úrovni tímu.

Agilná metóda Scrum používa tabuľka spálených hodín tzv. „burn-down chart“[6]. Začína sa s počtom odhadovaných hodin na iteráciu a každý za seba ako pracuje, postupne znižuje počet odhadovaných hodín na úlohách. Takto je vidieť celkový počet hodín, ktorý by sa mal už odpracovať ako aj odhadovaný počet hodín do konca. Správne by sa iterácia mala skončiť na nule (viď Obr. 1.).



Obr. 1. Scrum burn-down chart

Úlohy sú hotové až keď sú hotové

Možno to bude znieť ako intuitívne jasné, ale hlavne pri práci na softvérových projektoch môžeme od programátorov počuť, že majú niečo hotové, avšak keď po nich chceme, aby to predviedli môže sa stať, že povedia, že ešte musia dokončiť toto alebo toto.

Dodržanie tohto kroku je podľa môjho názoru kľúčový atribút na to, aby plánovanie malo aspoň nejakú šancu „trafiť sa“ do reálneho stavu, ktorý nastane na konci. Pre programátorov je veľmi lákavé začať pracovať na projekte do výšky a následne doťahovať do šírky. Takýto prístup je ale z hľadiska plánovania nevhodný, pretože sa môže stať, že projekt postihne syndróm 90% hotovo [1] a na konci iterácie sa zistí, že skoro nič nie je hotové na 100% a plán tak nie je dodržaný. Treba sa teda striktne držať pravidla, čo nie je na 100%, ako keby ani nebolo začaté [8].

Pri agilnom plánovaní sa plánuje iba pre danú iteráciu?

Toto nie je pravda a pokiaľ sa v agilnom projekte plánuje iba takto, je to znak nedostatočného plánovania. Plánovať treba vždy na troch úrovniach [2] – plán vydania (tzv. „release plan“), plán na iteráciu (tzv. „iteration plan“) a plán na aktuálny deň (tzv. „daily plan“), pričom každý plán sa robí s rôznym stupňom presnosti. Mať k dispozícii plány s rôznym časovým horizontom a rôznymi úrovňami presnosti má dve hlavné výhody.

V prvom rade dobre komunikuje fakt, že rôzne plány sa vypracúvajú z rôznych dôvodov. Denný plán, ktorý sa vytvára na dennom stretnutí, je relatívne presný: jednotlivci si vyberú, ktoré úlohy počas dňa dokončia alebo že pokročia na nejakej väčšej úlohe. Plán pre iteráciu je menej presný. Je to zoznam „používateľských príbehov“, na ktorých sa bude pracovať počas iterácie a úlohy s nimi spojené. Pretože v každom „používateľskom príbehu“ je istá dávka nepresnosti, je takisto nie celkom presne definované, čo znamená, že daný príbeh bude dokončený v danej iterácií. Plán vývoja je najmenej presný, keďže obsahuje iba prioritný zoznam „používateľských príbehov“ a jeden alebo viac odhadov koľko z požadovanej funkcionality je pravdepodobne možné dokončiť do dátumu vydania.

Druhá výhoda plánovania na rôznych úrovniach je, že pomáha tímu vidieť projekt z rôznych uhlov pohľadu. Plán pre iteráciu je nutný na koordinovanú prácu v tíme s rôznymi funkciami v rámci krátkej iterácie. Plán vydania ponúka širší pohľad na projekt a pomáha tímu nestratiť sa. Tím, ktorý pracuje z iterácie na iteráciu bez toho, aby si uvedomoval vzdialenejšie ciele, je ohrozený tým, že bude sledovať iba krátkodobé ciele a nebude brať ohľad na dlhodobý cieľ.

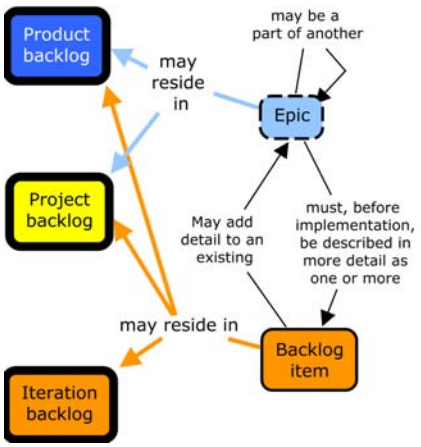
Pri tomto bode by som sa rád pozastavil, keďže ide podľa môjho názoru o jeden z najväčších rizík agilného vývoja softvéru. Ak projekt, ktorý sa takto vyvíja, nie je nutné zaradiť do dlhodobejšieho portfólia organizácie, resp. nie je súčasťou väčšieho softvérového projektu, tak na toto nie je potrebné myslieť. Avšak my nežijeme v ideálnom svete a skoro v každom prípade vyvíjame softvér s tým, že je buď súčasťou nejakého väčšieho celku, alebo musí minimálne spolupracovať s nejakou už

existujúcou infraštruktúrou. V tomto prípade je nesmierne dôležité, aby tím nestratil prehľad o dlhodobých cieľoch.

Agilné plánovanie v kontexte väčších projektov

Väčšina iteratívnych a inkrementálnych metód vývoja – ako napríklad agilné metódy – kladie dôraz na vytváranie softvéru vhodného na nasadenie v krátkych a fixných časových úsekoch. V praxi môže slabo organizovaný a rýchly vývoj s mnohými tímami viesť k fragmentácií a celkový obraz o práci a jej spojitosť s celkovou stratégiou firmy sa môže zahmliť. Dokonca aj v prípade, že sa pri plánovaní myslí na dlhodobé ciele, je tu stále reálna šanca, že sa v konečnom dôsledku prehliadnu v množstve „používateľských príbehov".

V rámci výskumného projektu na technickej univerzite v Helsinkách [7] bol implementovaný systém, ktorý by mal tento problém pomôcť vyriešiť. Systém čiastočne upravuje Scrum[6] metodiku a okrem zoznamu úloh pre daný šprint tzv. „Sprint backlog", ktorý nazvali zoznam úloh pre iteráciu tzv. „Iteration backlog" používajú aj zoznam úloh pre projekt tzv. „Project backlog" a zoznam úloh pre projekt tzv. „Product backlog". Normálna úloha tzv. „Backlog item" sa môže nachádzať vo všetkých úrovniach, pričom v zozname úloh pre projekt aj produkt môžu vystupovať aj globálne úlohy tzv. „Epics", čo sú práve akési vízie manažmentu, ktoré majú pomôcť splniť biznis plán (viď Obr. 2.).



Obr. 2. Návrh globálnych úloh v zozname úloh pre produkt.

Podľa môjho názoru ide o výrazné zlepšenie existujúcich metód, ktoré má ozajstnú šancu prispieť k tomu, aby sa pri práci na projektoch nezabúdalo na to, že vyvíjame produkt a nie iba danú iteráciu. Rieši tak tento problém veľmi elegantne a systémovo, keďže dramaticky nemení existujúcu štruktúru, na ktorú už programátori môžu byť zvyknutí.

Ako zohľadniť zmeny?

Medzi najčastejšie problémy, na ktoré softvérové projekty narážajú, nie sú len v plánovaní alebo neochote pracovať. Najväčšie problémy spôsobujú neustále sa meniace požiadavky zo strany zákazníka [4].

Zákazník často nevie, čo vlastne potrebuje, a iba vo veľmi výnimočných situáciách je spokojný s tým, čo povedal, že chce. Preto až keď sa mu predvedie prvý prototyp, si uvedomí, aké reálne požiadavky na systém vlastne má. To ale už býva z hľadiska dlhodobého plánovania neskoro a v klasických projektoch, kde sa agilné metodiky nepoužívajú, môže mať za následok neúspech projektu.

Agilné metodiky sa tomuto stavu snažia predísť neustálym kontaktom so zákazníkom, avšak aj tu sa dajú identifikovať nedostatky. Zákazník na začiatku súhlasí s požiadavkami na iteráciu a tie sú už nemenné. Zákazník môže obetovať jednu z požiadaviek za inú, prípadne ju pridať do požiadaviek ktoré sa implementujú iba ak zostane čas.

Správne plánovanie umožní pridať do celkových požiadaviek pre projekt aj novú funkčnosť bez toho aby sme ovplyvňovali normálny spôsob agilného riadenia projektu. Kľúčom k úspechu je v tomto prípade časté plánovanie po každej iterácií. V klasických projektoch by takéto niečo bolo nemožné. Avšak v agilných projektoch máme v oblasti plánovania výrazne navrch.

Táto výhoda ale prestavuje jedno z najväčších rizík [4]. Zmeny sa dajú rozdeliť do dvoch kategórií. Prvá kategória zmien je taká, ktorá sa nedá predvídať a je len na tom či sa dostane dostatočne vysoko, aby sa implementovala. V tejto kategórií sa veľa, z hľadiska plánovania, nedá urobiť. Druhá kategória ale predstavuje zmeny vyplývajúce z toho, že sa niektoré požiadavky nepresne špecifikovali a naplánovali. Vychádzalo sa totiž z toho, že v agilnom vývoji aj tak veci vyjasnia. Táto kategória predstavuje riziká priamo spojené s agilným vývojom a pokiaľ sa plánovaniu a špecifikovaniu nevenuje dostatočná pozornosť, môže to viesť k navýšeniu zdrojov potrebných na ich uskutočnenie. Jediným riešením je venovať pozornosť všetkým požiadavkám a poriadne ich špecifikovať hneď na začiatku.

A čo pracovníci?

Na celú problematiku plánovania ale existuje aj iný pohľad. Dalo by sa povedať, že ide o ten najdôležitejší, ide totiž o pohľad samotných pracovníkov/programátorov. Tí by v si ideálnom prípade mali vyberať úlohy sami a vypracovávať ich načas, ale v reálnych situáciách toto nie vždy funguje. Navyše ak nie sú dobre naplánované jednotlivé iterácie, môže dôjsť k problémom súvisiacim so stratením rytmu a následným problémom s dodržaním termínov. Za najdôležitejšiu vec, na ktorú treba myslieť pri práci na agilnom projekte, je že sa treba čo najviac snažiť držať pravidiel, ktoré tieto metódy hlásajú. Ide možno o triviálnu vec, ale nasledujúce riadky vás presvedčia o opaku.

V rámci štúdie vypracovanej na Sofijskej univerzite [5] sa preukázalo, že agilný vývoj má značné nedostatky ak sa presne nedodržujú metodiky. V troch tímoch sa snažili implementovať agilné metodiky pri vývoju softvéru.

Konkrétne išlo o metodiky Scrum [6], v ktorej sa vyvíja v presne zadelených úsekoch (šprintoch) trvajúcich spravidla 2-4 týždne. Pred nimi sa uskutoční tzv. predšprintový míting, kde sa zadefinuje, čo sa bude diať v nasledujúcom šprinte. Počas šprintu sa každý deň stretne tím v rovnakom čase a na rovnakom mieste na tzv. Scrum mítingu, ktorý spravidla netrvá viac ako 15 minút. Tu sa zadefinujú úlohy pre každého člena na nasledujúci deň tak, aby na ďalšom mítingu mohol referovať, čo spravil, prípadne kde nastali problémy.

Metodika Scrum je ale pripravená pre tímy vo firmách, kde sa pracuje 8 hodín denne 5 dní v týždni. Tieto projekty sa však realizovali na pôde univerzity pracovali na nich študenti denného štúdia, takže táto metóda bola mierne modifikovaná. Namiesto denných SCRUM mítingov sa konali týždenné mítingy.

Z týchto zmien vyplynulo mnoho problémov, ktorým by sa dalo vyhnúť. Za najväčší problém môžeme považovať fakt, že nedodržali pravidlá metodiky Scrum úplne presne. Z viacerých názorov vyplynulo, že keby sa presne držali metódy Scrum, projekty by boli omnoho úspešnejšie [3].

Ďalší problém, ktorý identifikovali bol nedostatok plánovania. Tímy sa vyjadrili, že lepšie plánovanie by zlepšilo prácu na projekte. V neposlednom rade hrá rolu aj to, že tímy stratili rytmus kvôli prestávke počas prázdnin. Toto by sa dalo opäť pripísať nesprávnemu plánovaniu a naplánovať túto prestávku lepšie by taktiež prispelo k úspešnejšiemu projektu. S plánovaním priamo súvisia aj menej časté Scrum mítingy, ktoré výrazne zlepšujú komunikáciu a plánovanie na dennej báze v projekte.

V neposlednom rade sa všetky tímy zhodli na jednom nedostatku a tým bolo, že nemali k dispozícií priestor, kde by mohli pracovať spoločne. Keďže komunikácia je jedným z hlavných pilierov pri agilných metódach, plánovanie v takomto prostredí bolo o to ťažšie. Tento fakt podporený tým, že tímy nemali dovtedy žiadne skúsenosti s agilnými metódami a plánovaním v takomto prostredí, výrazne prispeli k tomu, že projekty sa nakoniec po druhom šprinte odovzdali, ale neboli bez problémov.

Záver

Na záver by sme sa mohli spýtať či sú agilné metódy a plánovanie v nich všeliek na problémy, ktoré sprevádzajú nielen vývoj softvéru? Odpoveď môže znieť alibisticky, ale áno aj nie. Ani ten najlepšie pripravený plán nemôže počítať so všetkými alternatívami, či už pri klasickom alebo agilnom plánovaní. A keď nič nepredvídateľné nenastane, môže neúspech spôsobiť jednoduché nedodržanie príp. nepochopenie agilných pravidiel.

Napriek tomu, že agilné plánovanie má oproti klasickému plánovaniu veľa výhod (ako napr. plánovanie na základe vlastností produktu a nie úloh, a pod.), nepredstavuje mýtickú „striebornú guľku“ [1]. Predstavuje ale kvalitnú a oveľa robustnejšiu alternatívu ku klasickému plánovaniu, ktorá keď sa správne používa, dramaticky zlepší vývoj v projektoch hlavne preto, že sa počíta s nepredvídateľnosťou vývoja.

Použitá literatúra

1. Brooks, F.P., Jr.: The Mythical Man-Month: Essays on Software Engineering (Anniversary Ed.). Addison-Wesley Longman Inc., Boston, 1995. ISBN 0201835959
2. Cohn, Mike. Agile Estimating and Planning. s.l. : Prentice Hall.
3. Cohn, Mike. Toward a Catalog of Scrum Smells, Scrum Alliance, október 2003. Dostupné z http://www.mountaingoatsoftware.com/system/article/file/11/ScrumSmells.pdf [citované 17.10.2008]
4. Elke Hochmüller: Agile Process Myths, ACM Press, máj 2008, s. 3.
5. Krasteva, Iva: Adopting an Agile Methodology — Why It Did Not Work, ACM Press, máj 2008
6. Schwaber, Ken. Agile project management with Scrum. : Microsoft Press. 2004. ISBN 0-735-61993-x
7. Vähäniitty, Elke: Towards a Conceptual Framework and Tool Support for Linking Long-term Product and Business Planning with Agile Software Development, ACM Press, máj 2008, s. 26
8. Kolektív autorov: Agile Principle #7: "Done" Means "DONE!". Dostupné z http://www.agile-software-development.com/2007/04/agile-principle-7-done-means-done.html [citované 17.10.2008]

Annotation

Agile planning

Planning is one of the most important parts of a software project. Managing of software projects has been open to new methods such as agile and/or extreme programming. In such project planning is not the primary concern and is used primarily to establish goals while the work schedule is left to the workers themselves. Many projects that use agile methods can benefit from better planning. The goal of this essay is to show risks when using agile methods and to offer alternatives to improve using better planning.