MALÝM STAČÍ ŠPRINT, VEĽKÝM TREBA DVA PLÁNY

Nie je čas plytvať časom.

Ivana Bohunická

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

Abstrakt. *Plánovanie je nevyhnutnou súčasťou tvorby softvérového projektu a určuje jeho smerovanie od samotného počiatku. Vytvorený plán sa vezie s projektom celý jeho život, preto je potrebné pripraviť skutočne dobrý plán. V eseji chcem pripomenúť, že vytvorenie plánu nie je jednoduché a prináša viacero problémov. Našťastie existuje množstvo stratégii ako sa k plánovaniu postaviť. Zaujímavým postupom je spôsob dvoch plánov. Jeden predstavuje pesimistický plán určený pre zákazníka a druhý, optimistický plán, určený pre vývojárov. Ďalší prístup, ktorému budem venovať značnú pozornosť, je agilný vývoj a jeho metóda Scrum. Scrum je orientovaný na ľudí a ich vzťahy, procesy sú v tejto metóde druhoradé. Snahou tejto eseje je zhodnotiť tieto dva prístupy, vzájomne ich porovnať a odporučiť ich použitie v závislosti od druhu projektu.*

Kľúčové slová: *plánovanie vývoja softvéru, problémy plánovania, agilný vývoj, Scrum, dva plány*

Úvod

Plánovanie má mnohé úskalia a problémy. Zistila som to pri písaní tejto eseje prakticky na vlastnej koži. Naplánovala som si prácu na eseji, ale nepredpovedala a nezapočítala som všetky riziká, ktoré môžu nastať. Písanie eseje nešlo totiž úplne podľa mojich predstáv, s prvým výsledkom som nebola spokojná, aj keď mi zabral veľa času.

Našťastie sa nič hrozné nestalo, pretože som začala dostatočne skoro a môj globálny plán pre všetky predmety pripúšťa aj takéto situácie. Aspoňže tento plán nezlyhal! Otriasla som sa z nezdaru a následne mohol vzniknúť tento úprimný úvod.

Je zrejmé, že plánovanie je náročné a problematické. Obzvlášť svoje špecifiká má plánovanie vývoja softvéru. Vychádzam z tejto informácie a hľadám pre plánovanie vhodný prístup, aby sa znížil rozsah problémov a rizík. Súčasne každý softvérový produkt je jedinečný a vyžaduje rozličný prístup k tvorbe plánu jeho vývoja.

Kde je problém?

Pozrime sa najskôr na problémy a odchýlky, ktoré ovplyvňujú plány. Philip G. Armour vidí zdroj odchýlok plánov vo faktoroch, ktorými sú: rozsah projektu, výkonnosť tímu, obchodný trh a technológie [1]. Rozsah projektu, pretože úplne na začiatku je problematické odhadnúť rozsah projektu a všetku funkcionalitu, ktorú bude produkt vykonávať. Ďalším faktorom je výkonnosť akou budeme v tíme schopný získavať požadované znalosti. Toto nevieme dobre predpovedať a najmä nie ovplyvniť. Odchýlky môže spôsobovať aj vývoj na obchodných trhoch, pretože dlho tvorený produkt môže pri vstupe na trh byť už zastaraný. Posledným faktorom sú technológie, ktoré sa neustále menia a aj to môže mať vplyv na vývoj projektu. Pri technológiách by som sa pozastavila vo väčšej miere, pretože na rozdiel od ostatných uvedených faktorov, technológie sú výlučne špecifické práve pre plány ohľadom vývoja softvéru.

Rozpoviem stručne príbeh ako technológia ovplyvnila projekt, na ktorom pracujem v praxi. Pracujem na veľkom informačnom systéme implementovanom v Jave. Keď som nastúpila, zarazil ma príkaz, že si mám nainštalovať relatívne staršiu verziu Javy. Vysvetlenie však bolo jednoduché. Nové verzie prinášajú aj nové chyby. V našom projekte sa opravovali tieto chyby po určitú verziu a ďalej už nie. Nebol na to čas, prostriedky a ani vôľa. Technológii sme boli ochotní podriaďovať plány iba do určitej miery a do určitého času. Našťastie si to môžeme zatiaľ dovoliť.

Po analýze príčin, ktoré vedú k narušeniu plánov autor ďalej v práci ponúka isté riešenia [1]. Hovorí, že môžeme dúfať, že nám sa žiaden takýto problém nestane. Taktiež sa môžeme spoliehať na to, že počet negatívnych odchýlok sa vyrovná počtu pozitívnych, čím sa stav vyrovná. Ja však považujem tieto riešenia iba za akési strkanie hlavy do piesku. Nemôžeme sa naivne spoliehať na to, že sa nám problém nestane respektíve, že sa vyrieši sám. Dôkazom toho, že sa na takýto zázrak spoľahnúť nedá sú samotné firmy. Myslím, že práve uvedené faktory spôsobujú večné meškanie dodania softvérového produktu a odklady termínov, ktoré pozná väčšina firiem. Ako riešiť takúto situáciu?

Neplánovať nie je riešenie

„Každý, kto pracoval na zle plánovanom projekte vie, že neplánovať nie je odpoveď “ [1]. Poznám príklad z praxe, kedy sa podcenil menší softvérový projekt a nebol vytvorený preňho žiaden plán. Odovzdanie projektu sa neočakávane predĺžilo o dva mesiace.

Plánovanie je dôležité. Za pravdu mi dáva aj autor Steve McConnell, ktorý vtipne pomenoval nedostatky v plánovaní ako smrteľné hriechy [2]. Za prvý, azda najväčší hriech považuje, ak sa neplánuje vôbec. Ďalším, asi najdôležitejším je nedostatočné plánovanie.

Pre lepšiu ilustráciu opäť využijem zážitok z praxe, kde sme nešťastne doplatili na nedostatočné plánovanie. Architekt navrhol veľkú zmenu v projekte, vedelo sa čo treba spraviť, stanovila sa množina vývojárov, ktorá budú na tom pracovať, ale to bolo všetko.

Začalo sa nestíhať. Aktuálne fungujeme v krízovom režime, začali každodenné schôdze a plánovanie vo veľkom. Nazdávam sa, že to situácii pomohlo. Omnoho lepšie by však bolo, keby hneď na začiatku vznikne poriadny plán a mohli sme sa vyhnúť krízovému režimu.

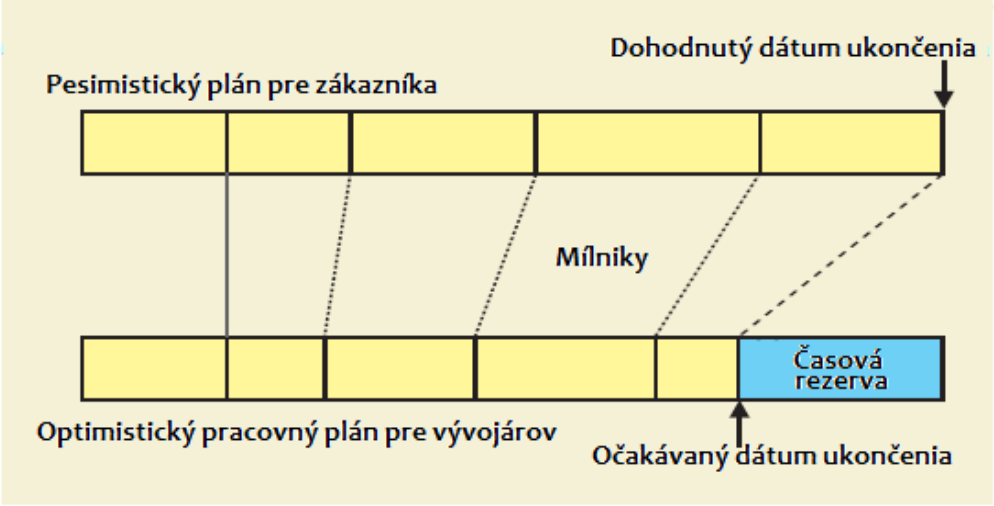
Stále som však neposkytla utešujúce riešenie problému. Vieme už, že vytvoriť plán je nutné. Ako však môžeme vytvoriť dokonalý plán?

Dva plány

V prvom rade sa musíme zmieriť s tým, že náš plán nebude nikdy dokonalý. Ja osobne vždy vravím, že každý ľudský výtvor, či už hmotný alebo nehmotný, má svoje hranice, obmedzenia a nemôže byť perfektný. Môžeme sa ale pokúsiť aspoň o zdokonaľovanie.

Problémy, ktoré nastávajú v plánoch, jednoducho nazvem katastrofický scenár. Našou úlohou je zahrnúť tento scenár do plánovania. V prvom rade s ním treba počítať. Zároveň však treba počítať, že nastať nemusí alebo nemusí nastať v celom rozsahu. A sme opäť pri pôvodnom pláne bez katastrof. Autor Philip G. Armour reaguje presne na tento stav, ktorý som načrtla, a ponúka prosté riešenie - dva plány [1].

Jeden plán má predstavovať katastrofický scenár, kde je potrebné zahrnúť všetky potenciálne odchýlky a riziká. Takýto plán, nazvem ho pesimistický, sa predloží zákazníkovi. Zároveň je však potrebné zachovať efektívne a stabilné pracovné tempo, preto potrebujeme ešte druhý plán. Druhý, optimistický plán, je predpokladaný pracovný plán, podľa ktorého postupujú vývojári. Takto sa zabezpečí to, že vývojári pracujú efektívne, a ak by sa aj niečo stalo, vždy je tu ešte prvý plán s katastrofickým scenárom. Myslím, že zároveň zákazník bude spokojný, pretože počíta takpovediac s najhorším a v skutočnosti môže byť aj milo prekvapený, ak sa projekt dokončí skôr. Pre lepšiu ilustráciu uvádzam obrázok (Obr.1.).



Obr. 1. Dva plány [1].

Keď som začala uvažovať nad realizáciou tohto návrhu v praxi, pousmiala som sa, nakoľko som rozmýšľala, ako predložiť túto stratégiu vývojárom. Nemôžu byť predsa informovaní o pesimistickom pláne, pretože v tom momente by sa stratilo čaro optimistického plánu. Vývojári by sa spoliehali na pesimistický scenár a pracovali by neefektívne a v pomalom tempe. Budú sa teda manažéri s vývojármi firmy hrať na skrývačku? V malom tíme asi ťažko! Práve preto vidím v tomto problém, ktorý sa ale nemusí týkať veľkých firiem. Vo veľkých firmách a veľkých projektoch, ktoré môžu byť aj medzinárodné, často manažéri a vývojári nekomunikujú priamo tvárou v tvár alebo ich vzťahy sú iba formálne. V tomto prípade sú schopní druhý plán doslova utajiť.

Myslím si, že tento spôsob plánovania vyžaduje viacero plánovačov a manažérov na jeho vytvorenie a udržiavanie, čo však pre veľkú firmu nemusí predstavovať problém. V tomto ohľade môže byť prístup dvoch plánov náročnejší, ale zároveň sa vďaka nemu jednoduchšie zabezpečí dodržiavanie plánov. Pre uvedené dôvody sa mi tento postup páči ako možnosť plánovania pre veľké medzinárodné firmy a spoločnosti. Takéto firmy sú neprispôsobivé voči zmenám a funguje v nich silná hierarchickosť medzi zamestnancami. Používajú zvyčajne jeden plán. Použitie dvoch plánov môže priniesť oživenie vývojového procesu, ako aj reálnejšie napĺňanie plánov.

Nedokonalosť tohto riešenia vidím v tom, že je nezmyselné mať hoci aj dva plány, ak sa aktuálny stav riešenia projektu vymkne spod kontroly oboch plánov. V princípe sa nemôžeme striktne pridŕžať jedného či druhého scenára. Napadá mi, prečo nepoužijeme tri plány alebo viacero plánov? Asi preto, že v toľkých plánoch by napokon vznikol chaos, a to by nebolo šťastné riešenie. Myslím, že riešením by bolo mať k dispozícii nejaké flexibilné mantinely ako súčasť jedného aj druhého plánu. Samozrejme plán je od toho, aby sa dodržiaval a tak sa dosiahol istý efekt. Nemôžeme sa však bezhlavo držať každého určeného kroku. Preto sa plán musí vyvíjať spolu s vývojom softvéru.

V tomto momente naznačujem akýsi benevolentnejší prístup k plánovaniu. Možno práve agilné plánovanie je spôsob, kam smerujú moje úvahy.

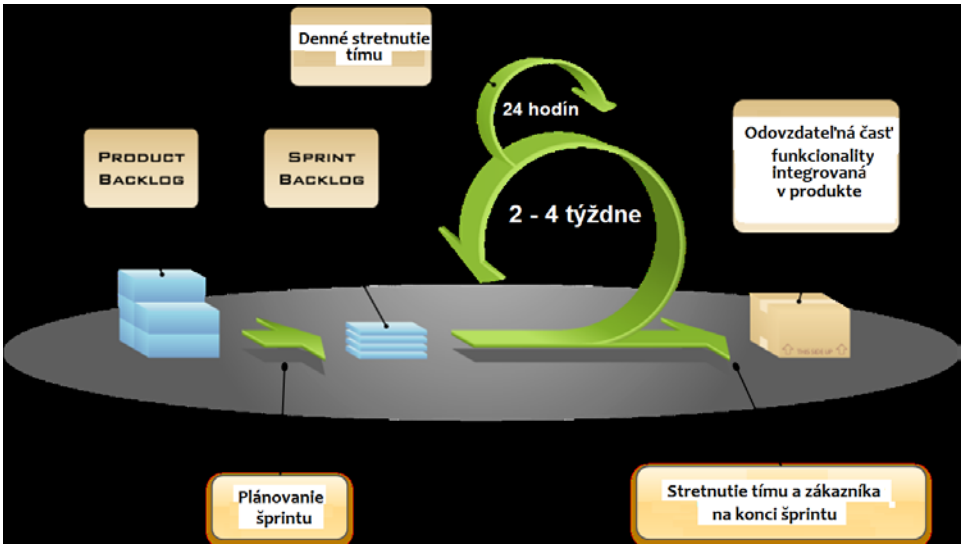
Plánovanie agilným spôsobom

Agilný vývoj softvéru najlepšie vystihuje a charakterizuje jeho manifest: Individuality a vzájomná interakcia je nad procesmi a nástrojmi. Fungujúci softvér je nad obsiahlou dokumentáciou. Spolupráca so zákazníkom je nad uzavretou zmluvou. Reakcia na zmeny je nad nasledovaním plánu [3].

Uvedomujem si, že pre študentov informatických odborov je náročné zabezpečiť, aby si vyskúšali manažment vývoja softvéru. „Študenti informatiky majú zriedka príležitosť praktizovať manažérske aktivity“ [4]. Našťastie existuje agilný vývoj, ktorý aj v školských podmienkach môže prebiehať na zodpovedajúcej úrovni. Agilný prístup k vývoju má viacero možností realizácie, napríklad metódu Scrum. Metóda Scrum je práve tá, ktorú si vyskúšame takpovediac na vlastnej koži, formou upravenou našim akademickým podmienkam.

V Scrum metóde sa na začiatku vývoja produktu stanoví zoznam požiadaviek na produkt, ktorý sa nazýva Product backlog. Vývoj Scrum metódou prebieha v krátkych a rovnako dlhých iteráciách nazývaných šprint. Do šprintu sa vyberajú úlohy z Product backlog, ktoré majú byť dokončené na konci šprintu, nazývajú sa Sprint backlog.

Výstupom každého šprintu musí byť časť funkcionality produktu, ktorá je integrovaná do produktu a pripravená na odovzdanie [5]. Počas iterácií prebiehajú viaceré stretnutia vývojového tímu, ako aj stretnutia tímu so zákazníkom. Tento proces najlepšie ilustruje nasledujúci obrázok (Obr.2.).



Obr. 2. Plánovanie metódou Scrum [6].

Počas práce v tímovom projekte Scrum metódou som si uvedomila, že tento postup môže viesť k chaosu. Je pravda, že Scrum môže navonok pôsobiť ako chaos, ale v internom fungovaní tímu musí byť poriadok. Samotný názov Scrum je pôvodne pojem z amerického futbalu, ktorý bol prebraný na označenie agilnej metódy. Pojem odkazuje na tím amerických futbalistov, ktorý si navzájom rozumejú a organizujú sa, aj keď navonok pôsobia chaoticky [6]. Presne takto navonok pôsobí aj vývojový tím pracujúci agilnou metódou Scrum. Dôležité je teda zachovať vo vnútri tímu organizáciu a poriadok. Riziková je v tomto ohľade neskúsenosť členov pracovať v tíme a neskúsenosť pracovať touto metódou, čo je aj náš problém v tímovom projekte. Neskúsenosť je však podľa mňa rizikový faktor pre akýkoľvek vývojový postup, či plán. Azda ešte väčší problém vidím v motivácii členov tímu. Ako uvádza autor [6] vo svojej práci, Scrum veľmi závisí na vysoko motivovaných, úzko spolupracujúcich, funkčných a samo organizovaných tímoch, pre ktorých sa plánuje skutočne jednoducho, dokonca až samovoľne. Študenti a členovia tímu musia byť motivovaný najmä preto, že je potrebné produkovať správnu a hotovú funkcionalitu vo veľmi krátkych intervaloch. V školských podmienkach je problém zoskupiť tím ideálne motivovaných študentov. Zvyčajne je výsledkom tím skladajúci sa z rôzne motivovaných, rôzne snaživých a prípadne aj nemotivovaných študentov. Preto si v reálnej praxi viem predstaviť fungovanie Scrum metódy iba vo vysoko motivovanom tíme. V takomto prostredí sa plány tvoria jednoducho, každý člen si korektne plní povinnosti vychádzajúc z podstaty motivácie. Ak by ale motivácia zmizla, aj plánovanie by zrazu bolo zložitejšie a nevravím už o dodržiavaní plánu, čím by mohol reálne v tíme vedenom Scrum metódou vzniknúť chaos.

Dva plány verzus agilné plánovanie

Agilné plánovanie vnímam ako možnosť veľmi zaujímavého a novšieho postupu plánovania. Vo všeobecnosti sa prikláňam k názoru, že agilný vývoj je vhodný pre malé tímy a malé projekty. Napríklad neformálna komunikácia sa jednoduchšie dosiahne v malom spoločenstve, ako vo veľkej firme, kde sú si ľudia vzájomne nadriadení, respektíve podriadení. V menšom kolektíve je možné, aby sa tím organizoval a manažoval sám, pričom je aj prípustné ak si zamieňajú role v tíme. Pre mňa nie je predstaviteľné, aby v malej firme alebo v malom tíme s piatimi až desiatimi ľuďmi, vykonávali dvaja výlučne manažérsku funkciu.

Takže naopak, scenár s dvoma plánmi vidím ako neštandardnú, ale vhodnú stratégiu pre veľké spoločnosti s veľkými tímami a projektmi. Vo firme so stovkami zamestnancov a množstvom projektov po celom svete je náročné prihliadať na záujmy každého jednotlivca a neustále meniť plány. Zároveň predpokladám, že tvorba a najmä údržba dvoch plánov si vyžaduje viacej ľudí a kapacít, čo je možné realizovať práve vo veľkých firmách s rozsiahlym manažmentom. Malé firmy často nemajú ani poriadny manažment. V prostredí veľkých spoločností je aj ťažšie dosiahnuť priateľskú a neformálnu atmosféru, najmä čo sa riadenia a manažmentu týka.

Záver

V závere mi jednoducho vyplýva, že existuje viacero možností plánovania vývoja softvéru. Žiadna možnosť nie je vyslovene zlá alebo vyslovene dobrá. Dôležité je zvoliť si správnu v závislosti od typu projektu. Každý projekt je iný, každý tím je iný, preto aj plánovanie musí byť rôzne. Našťastie existuje množstvo prístupov. Aj samotný agilný vývoj, ktorý som načrtla, má niekoľko metód realizácie. Scrum a aj ďalšie podobné metódy odporúčam pre menšie projekty realizované malými vysoko motivovanými tímami. Myslím, že veľkým tímom by zase mohla pomôcť stratégia dvoch plánov. Pomocou nej by sa aspoň čiastočne odpútali od robustných a ťažkopádnych plánov typických pre veľké projekty.

Použitá literatúra

1. Armour, P.G.: To plan, two plans. *Communications of the ACM*, 2005, vol. 48, no. 9, s. 15-19.
2. McConnel, S.: The Nine Deadly Sins of Project Planning. *IEEE Software*, 2001, vol. 18, no. 5, s. 5- 7.
3. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, C.R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: *Manifesto for Agile Software Development*, 2001, http://agilemanifesto.org/
4. Keenan, F.: Learning Project Planning the Agile Way. *Innovation and technology in computer science education*, 2006, vol. 38, no. 3, s. 324.
5. Fernandes, J.M., Almeida, M.: Classification and Comparison of Agile Methods. Quality of Information and Communications Technology, 2010, vol. , no. , s. 391- 396.

6. Hoda, R.: Self organizing agile teams: A grounded theory. Victoria University of Wellinton, 2011. Thesis.

Annotation

Sprint is enough for small, big required two plans.

Planning is necessary part of software development and defines project direction from its beginning. Created plan lives with project its whole life, so it is necessary to prepare really good plan. In the essay, I would like to remind, that plan creation it not easy and it brings a lot of problems. Fortunately, many strategies of planning exist. Interesting strategy is the way of two plans- one pessimistic plan for customer and second optimistic plan for developers. Another strategy is agile development and its Scrum method. Scrum is oriented on humans and their relationships, processes are low-grade in this method. Effort of this essay is to estimate these two methods, compare them and recommend them depending on project type.