PLÁNOVANIE ÚSPECHU

Plánovať je ako milovať. Keď ju miluješ, nie je čo riešiť!

Jozef Rešetár

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

Abstrakt. *Väčšina ľudí nerada pozerá do budúcnosti a radšej si užívajú prítomnosť, čo najviac ako môžu. Pričom nemyslia na to, ako sa to môže odzrkadliť na ich budúcnosti. Neplánujú, nechcú plánovať, pretože ich to môže o niečo v tomto momente pripraviť - minimálne o čas, ktorý pri príprave plánu alebo rozvrhu mohli stráviť. Neuvedomujú si, že plán je veľmi dôležitá súčasť každej práce a aj keď si plánujú jednoduché veci ako napríklad, čo dnes budú robiť, zložitejšie a častokrát oveľa dôležitejšie plány sa im vytvárajú ťažšie, odkladajú ich na neskôr až sa stane, že plán nepotrebujú lebo danú situáciu musia riešiť ihneď – zapochodu. Paradoxom je práve to, že nie plánovanie zabíja čas, ale neplánovanie je častým zdrojom opakovania veci z dôvodu pochybenia alebo celkového zlyhania. Nechuť k plánovaniu sa prejavuje aj vo vývoji softvéru. Táto skutočnosť ma vedie k téme tejto eseje – Plánovanie úspechu. Čo je pre úspešné plánovanie dôležité? Majú vplyv dobré odhady na plánovanie? Ako čeliť pričinám neúspechu pri vývoji softvéru z hľadiska plánovania? A nakoniec, prečo sme pochybili pri prvom odovzdaní tímového projektu? Odpovede na tieto otázky nájdete v tejto eseji.*

Kľúčové slová*: manažér, zákazník, vývojár, softvérový produkt, rozvrh, úspech, Ganttova schéma, metóda kritickej cesty, technika hodnotenia a revízie projektu*

Úvod

Nereálne alebo nejasné ciele, nepresný odhad požadovaných zdrojov alebo neschopnosť čeliť komplexnosti projektu [3] - to sú niektoré z častých príčin zlyhania pri vývoji softvéru. Konkrétne tieto tri veľmi úzko súvisia s plánovaním a tvorbou rozvrhu.

Počas práce na tímovom projekte sa nám stalo, že sme nestihli splniť plán šprintu, teda to, čo sme si zaumienili pred jeho začatím. Stalo sa to kvôli tomu, že sme nemali realistický odhad, koľko nám budú dané úlohy trvať. Namiesto 10 – 12 hodín sme si určili nerealistický odhad ukončenia úloh na 2 – 3 hodiny. Tým pádom, že úlohy boli z časového hľadiska jednoduché, nechali sme si ich riešenie na poslednú chvíľu – „Veď to nebude trvať dlho! Je to banalita!“, povedali sme si. No tri dni pred odovzdaním prišli komplikácie pri implementácii (s tými 2-3 hodinovými úlohami), s ktorými nikto z nás nepočítal a kompletizácia našich úloh sa veľmi predĺžila. Úlohy na seba nadväzovali a aj keď niektorí svoje úlohy stihli dokončiť včas, tí, ktorí mali na ich prácu nadviazať ani nezačali. Termín uplynul a my sme prezentovali vedúcemu projektu polovičatý produkt šprintu. A začalo to tak nevinne – pri rozhodovaní koľko približne času nám jednotlivé úlohy zaberú.

Bolo to len prvé odovzdanie, nie odovzdanie finálneho produktu za zimný semester. Našu známku to ovplyvní menej ako neúplné odovzdanie finálnej verzie zimného semestra, ale bolo by od nás veľmi nezodpovedné chybu neanalyzovať a nepoučiť sa z nej čo najviac, ako sa dá, aby sa zas nezopakovala.

Čo sa týka projektov ako takých tak, jedna štúdia úspešnosti ukončenia projektov preukázala, že len 30% projektov bolo ukončených v rámci odhadov a len 8% z nich vytvorilo odhady líšiace sa od skutočnosti do 10% [2]. Tieto čísla naznačujú ako veľmi je potrebné venovať sa plánovaniu a odhadom. Zákazníkov určite nepoteší, keď sa produkt omešká čo i len o jeden mesiac.

Rôzne odhady a vplyv na plánovanie

Jednou z príčin zlyhania softvéru, ktoré som na začiatku uviedol sú nepresné odhady. Vo všeobecnosti platí, že čím je odhad presnejší, tým je lepší. Presnosť je dôležitá pri tvorbe rozvrhu a preto v prípade nepresných odhadov, sú vytvárané plány alebo rozvrhy, ktoré sa nedajú realizovať z dôvodu, že sú častokrát veľmi priaznivé pre zákazníka, ktorý chce mať produkt na stole čím skôr a naopak nepriaznivé – a z toho dôvodu nezrealizovateľné – pre vývojárov, ktorí by potrebovali na jeho vypustenie v danom termíne štyridsaťosem hodinový deň.

Techniky ako robiť takéto odhady sú rôzne – techniky založené na expertnom usudzovaní, na formálnych modeloch, na analógii a techniky, ktoré sú kombináciou predchádzajúcich. V našom prípade sa analógia použiť nedala. Nemali sme predtým skúsenosti s vývojom takéhoto typu softvéru a podľa môjho názoru každý projekt je jedinečný, preto by som túto metódu bral s rezervou, kedže je založená na precedense úspešne ukončených projektov a veľký význam by som jej neprikladal, pretože je tam možnosť, že manažér, ktorý bude odhadovať podľa už úspešného projektu bude veľmi subjektívny a tým skreslí pohľad na nový projekt. Formálne modely ako napríklad COCOMO, analýza funkčných bodov, analýza prípadov použitia by nám takisto nepomohli, pretože naša znalosť problematiky bola v čase plánovania šprintu zanedbateľne malá – nevedeli sme ako ľahko sa, čo dá implementovať. Mali sme síce dokumentáciu na vysokej úrovni, ale problém bol hlavne v implementácii v jazyku, v ktorom nikto z nás doposiaľ neprogramoval. Bolo bádať potrebu hlbšej analýzy prostredia, ale nato sme nemali dostatok času, čo bolo spôsobené, podľa môjho názoru,

neskorým pridelením tímového projektu. Jediné, čo nám ostalo bolo expertné usudzovanie. Kedže tímový projekt vyvíjame metódou Scrum, tak v tomto prípade expertným usudzovaním bol plánovací poker. No pre nedostatok skúseností v tejto oblasti naš tím pri odhadovaní zlyhal ako bolo opísané vyššie.

Plány sa podľa situácie vo vývojárskom tíme neustále menia a prispôsobujú okolnostiam, ktoré vznikajú. Dá sa povedať, že plánujeme nato, aby sme na ďalšom stretnutí daný plán zahodili a podľa okolností a na základe starého plánu vytvorili ten nový. Tu je namieste nasledujúca otázka. Oplatí sa teda plánovať projekt, jeho jednotlivé šprinty? Určite áno! Dôvodov je viacero. Tvorením plánu sa redukuje neurčitosť projektu, hoci sa plány neustále menia. Treba povedať, že nemení sa práca, ale upravuje sa rozvrh, prerozdeľuje sa práca tak, aby sa všetko stihlo do zadaného termínu. Ďalším dôvodom je, že vývojari týmto v hrubých črtoch začínajú hlbšie chápať celému projektu a zmyslu, prečo niektoré veci musia robiť tak zložito. S týmto jednoznačne súhlasím, nevedel by som si predstaviť programovať na projekte, o ktorom by som vedel jedine so svojich úloh. Nebavilo by ma to. Samozrejme úzko súvisiace s týmto je aj zlepšenie výkonu, kde vývojár je motivovaný k lepším úspechom.

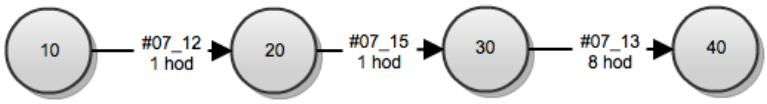
Na druhej strane z pohľadu manažéra má manažer v rukách nástroj, ktorým dokáže riadiť a sledovať, či sa projekt blíži k cieľom v stanovenom termíne alebo treba konať, ešte kým je čas a stimulovať projekt tam, kde to je potrebné. Tento nástroj – plán – v rukách manažéra dokáže včas odhaliť, že sa na projekte nepracuje tak, ako by sa malo.

Manažér je človek stojaci medzi vývojármi a zákazníkom, ktorý sa snaží nájsť kompromis medzi týmito dvoma tábormi, hlavne čo sa týka odovzdania produktu. Najväčšou zbraňou každého dobrého manažéra je navrhovať a hľadať riešenia [1]. Autor predchádzajúcej vety hovorí o príklade s loďou. Chcete si kúpiť novú loď, ale nemáte nato peniaze. Rozhodnete sa predať vaše auto. Nato aby ste mali novú loď, potrebujete predať auto za 5000$. Našiel sa kupec, ktorý ale ponúka len 4500$. Ak nato nepristúpite lebo chcete svojich 5000$ dolárov, aby ste mali na loď, nedostanete ani peniaze ani tú loď, lebo iný kupec sa nenájde. Ale zamyslite sa, čo je vaším cieľom, 5000$ alebo tá loď? Loď. Tak hľadajte so záujemcom o vaše auto spoločné riešenia! Zistíte, že je významným dodávateľom lodí vo vašom okolí, a že je vám schopný urobiť cenu tak, že na tom ešte 1000$ zarobíte. Zamerajte sa skôr na záujmy ako na pozíciu, v ktorej sa aktuálne nachádzate. Potrebuje zákazník celý produkt do konkrétneho dátumu? Nedá sa produkt rozdeliť na viac častí? Ak zákazník potrebuje naimplementovanú danú vlastnosť, potrebuje ju mať naimplementovanú zo všetkými detailami?

Čo sa týka toho dátumu, kedy si zákazník predstavuje hotový produkt na svojom stole. Už intuitívne cítime, že zákazník bude vo väčšine prípadov požadovať produkt stále skôr, ako je to vôbec možné urobiť. S tým, že si treba nechať vždy poradiť od vývojárov, ktorí manažérovi hovoria, že to sa nedá v danom termíne vyriešiť, s autorom súhlasím len polovične. Je pravdou, že vývojári majú väčšinou pravdu, ale je tu možnosť, že manažér dostal na krk bandu vývojárov, ktorým sa implementovať veľmi nechce. V tom prípade musí mať manažer zdraví sedliacky rozum, odhaliť to a využiť všetky dostupné prostriedky, aby ich dostatočne motivoval k výkonom, ktoré sa od nich v práci očakávajú. Teda aby neposlúchol odhady vývojárov, ale radšej sa, v takomto výnimočnom prípade, priklonil na stranu zákazníka.

Rozvrhy, ktoré sú kritické

Motiváciou, prečo projekt môže zlyhať je okrem nepresných odhadov neschopnosť čeliť komplexnosti problému, čo znamená, že problematika je tak veľká, že si ju ťažko predstaviť a vidieť všetky jej tajomné zákutia. Možným riešením je vizualizácia jej komplexnosti (napríklad Ganttovým diagramom) a keď zájdeme ešte hlbšie naznačením závislostí medzi jednotlivými úlohami metódou kritickej cesty. Medzi hlavné výhody použitia Ganttového diagramu patrí najmä explicitné a lineárne zobrazenie času ako aj termínov ukončenia úloh a projektu a zobrazenie podielov práce jednotlivých úloh, čo však v našom prípade nestačí. Podľa môjho názoru sa sledovanie podielu práce jednotlivých úloh v praxi sleduje ťažko. Vychádzam zo skúsenosti z tímového projektu, kde sa postup úlohy častokrát zaznačuje iba na konci, teda keď vývojár úlohu dokončí. Presne sledovať postup úlohy a podľa toho odhadnúť kedy sa daná úloha skončí je prakticky nemožné. Lebo počas nej sa môžu vyskytnúť komplikácie, ktoré vývojár častokrát nepredvídal a úloha sa zasekne na dlhší čas na nejakom percente vývoja.



Obr. 1. Kritická cesta.

Pretože schéma neukazuje závislosti jednotlivých úloh, ktoré sú na sebe závislé je prakticky pri riešení veľkých projektov nepoužiteľná. Takisto je nepoužiteľná kvôli neprehľadnosti pri väčšom počte úloh. Človek musí úlohy v systéme rolovať na obrazovke, aby si prešiel aký pokrok je v každej úlohe. Metóda kritickej cesty a technika hodnotenia a revízie projektu nehovorí o aktuálnom stav prác v úlohách, ale zato ukazuje zväčša na veľkých projektoch dôležité závislosti medzi úlohami. Na obrázku (pozri Obr. 1) je vidno konkrétny prípad, ktorý bol v našom tíme. Konkrétne tento prípad skončil neúspechom, kvôli uviaznutiu predchádzajúcich úloh.

Podľa môjho názoru by bolo zaujímavé prepojenie Ganttovho diagramu spolu s metódou kritickej cesty a techniky hodnotenia a revízie projektu a to takým spôsobom, aby sme z jedného diagramu vedeli prečítať, ako stojí s prácou na úlohe človek, na ktorom závisia ďalšie úlohy ďalších ľudí. Ak by úlohy na sebe nezáviseli, diagram by vyzeral úplne ako Ganttov diagram. Čo sa týka nevýhody, ktorú som spomínal pri Ganttovom diagrame s rolovaním, tak by sa z veľkej časti vyriešila tým, že úlohy by neboli pod sebou ale aj vedľa seba, čím by sa ušetril priestor. Samozrejme to nie je pravidlom.

Poslednou rozoberanou príčinou prečo vývoj softvéru zlyháva je nejasný alebo nereálny cieľ. Toto vo všeobecnosti znamená, že projekt potrebuje ako navrhuje autor článku [1], byť dodávaný zákazníkovi postupne vo viacerých verziách, čo v konečnom dôsledku vďaka Scrumu robíme. Pričom ale pripomína, že tie najdoležitejšie funkcie nech sú v najskorších verziách produktu. S tým s autorom súhlasím. Dá sa povedať, že takto zatiaľ nepostupujeme v našom projekte a možno aj kvôli tomu, sa nám nepodarilo šprint dokončiť včas. Kód sa posledné dni pred odovzdaním rapídne menil, štruktúroval do

abstraktnejších rovín, aby sa dalo potom na neho jednoducho nabaliť konkretizácia v podobe grafiky, zvukov, hernej logiky a podobne.

Autor ďalej hovorí, že vlastnosti, ktoré spolu súvisia treba dať do rovnakej verzie, takisto vlastnosti nemusia byť implementované do úplných detailov, ktoré sa aj tak používajú málokedy. V závere autor navrhuje zopár rád, ktoré môžu pomôcť pri zlepšovaní naplňaní plánu. Ak je to potrebné tak napríklad aj pridanie vývojárov nie je na škodu, ale varuje aby boli do projektu prizvaní zavčasu, lebo inak to spôsobí tzv. mýtický človeko-mesiac. To sa deje vtedy, keď si myslíme, že nám ďalší vývojár tesne pred odovzdávaním projektu urýchly jeho odovzdanie, ale on ho naopak spomalý, pretože sa musí s danou problematikou zoznámiť konkrétnejšie, čo zamestnáva iných vývojárov, ktorí mu to vysvetľujú. Ako motiváciu pre zlepšenie dodržiavania plánu určite prospejú súkromné kancelárie pre vývojárov, rýchlejšie počítače a iné veci, ktoré stimulujú pohodu vývojára. S týmto posledným bodom sa stotožňujem najviac. Napríklad spoločnosť Google je veľmi žiadaná práve týmito zamestnaneckými výhodami. Tým, že veľa vývojárov túži po práci v Google, môžu si manažéri spoločnosti dovoliť na svojich zamestnancov robiť väčší nátlak – zhusťovať im rozvrh práce - a tým ich pracovné výsledky musia narastať inak ich nahradí niekto iní, šikovnejší so záujmom pracovať pre takú spoločnosť akou je Google.

Záver

Ak mám zhrnúť odpovede na otázky položené v abstrakte, tak prvé dve si odpovedajú navzájom. Pre úspešné plánovanie sú dôležíté presné odhady, ktoré sa získavajú rôznymi spôsobmi, často ten najspoľahlivejší je praxou. Príčiny neúspechu pre plávanovanie sú tri: nepresné odhady, neschopnosť čeliť komplexnosti, ktorá sa rieši vizualizáciou problému a nejasné alebo nereálne ciele, ktoré rieši manažér často len komunikáciou so zákazníkmi a vysvetľovaním im, že to takto nejde. Naše tímové pochybenie bolo spôsobené viacerými faktormi – slabá analýza prostredia, v ktorom všetci po prvýkrát pracujeme, nerealistické odhady pri plánovaní šprintu a nakoniec osobný manažment času, kedy sme si úlohy odkladali na poslednú chvíľu, lebo podľa odhadov nám nemali trvať dlho. Zaujímavým riešením je prepojenie Ganttovej schémy s metodóu kritickej cesty, ktoré by bolo veľmi zaujímavé sledovať aké by toto zlepšenie malo výsledky pri reálnom nasadení. K úspešnému plánovaniu neodmysliteľne patrí aj správna motivácia. Pekným príkladom spoločnosti, ktorá sa stará o zamestnancov je Google, ktorá si vďaka zamestnaneckým výhodam môže dovoliť robiť plány, ktoré by normálna firma uskutočniť nemohla, pretože v nej nevládna taká morálka ako tam.

Použitá literatúra

1. McConnel, S.: How to defend an unpopular schedule. In: IEEE Software, Vol. 13, No. 3, 1996, 118-119
2. Jorgensen, M., Boehm, B.: Software Development Effort Estimation: Formal Models or expert judgement? In: IEEE Computer Society, IEEE Software, Vol. 26, 2009, 14-19
3. Charette, R.N.: Why software fails. In: IEEE Spectrum, Vol. 42, No. 9, 2005, 42-49

Annotation

Successful planning

The majority of people dislike looking to the future and instead of this live the present as much as they can. They don't think what follows their present actions. They don't plan, it bothers. They don't want to plan because this moment is for them the most they have. They don't want to lose the moment they like because of stupid plan making. They don't realize that planning is very important part of everyday working and although they plan simple things like - what I am going to do today – this is the most they can do. More difficult and important plans are done by them very rarely. They postpone making difficult plans to time when plans is not in need. Then they have to solve problems on the fly. Paradox is that planning is not timekiller, but can safe lot of time when it is done. The disgust to planning is testified also in software development. This fact leads me to the theme of this essay – Success planning. What is important for successful planning? What type of impact have good estimates on planning? How to face causes of failure in software development in terms of planning and scheduling management? And finally, why we failed to submit first prototype of out application in team project subject? You can find answers to these questions in this essay.