EFEKTÍVNE PLÁNOVANIE ČASU

Plány sú ničím, plánovanie je všetkým.

(Dwight D. Eisenhower)

Radovan Kuka

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

Abstrakt. *Plánovanie v softvérovom projekte je činnosť, bez ktorej by sa asi máloktorý projekt úspešne ukončil. Hovorí sa, že je lepšie mať zlý plán, ako žiadny. Ale čo ak je plán dobrý, ale termíny sa napriek tomu nestíhajú? Často môže byť problém v časovom manažmente samotných vývojárov. Pri jeho vytváraní sa môžu inšpirovať metódami, ktoré používajú manažéri pri zostavovaní plánu celého projektu. Avšak nie všetky metódy sú vhodné. Predkladaná esej sa snaží poukázať nielen na dôležitosť plánovania, ale aj na paralelu, medzi tvorbou osobných plánov a plánov softvérových projektov. Taktiež navrhuje metódu, ktorá by mohla byť nápomocná pri odhadovaní času potrebného pre úlohy. Zjednodušuje a urýchľuje túto nie veľmi obľúbenú činnosť a mohli by ju oceniť hlavne vývojári práve pri tvorbe osobných plánov.*

Kľúčové slová: *časový manažment, osobný plán, plánovanie, odhadovanie*

Úvod

Predstavme si ideálny plán vývoja softvéru, kde manažérmi odhadnuté dĺžky trvania úloh sa blížia skutočnosti. Nie vždy to však môže znamenať, že vývoj ukončíme v stanovenom termíne. Tieto plány totiž nezohľadňujú individualitu jednotlivých členov tímu. Jeden vývojár môže tento plán naplniť do poslednej bodky, no inému, s rovnakými schopnosťami a skúsenosťami, to môže trvať o niekoľko dní dlhšie. Problémom často býva osobný časový manažment.

Psychologické štúdie poukazujú na to, že ľudia majú tendenciu odkladať zložitejšie úlohy na neskôr [3]. Málokedy si však uvedomujú, že sa im tieto úlohy kopia čím ďalej, tým viac. Nakoniec sa môže stať, že niektoré z odkladaných úloh nestihnú včas dokončiť, alebo ich odfláknu. Častým problémom tiež býva plytvanie drahocenného času na nepodstatné činnosti ako písanie súkromných emailov a surfovanie po internete.

Z osobnej skúsenosti viem, že týmto zlozvykom trpia najmä študenti po príchode do prvého zamestnania. Avšak sebadisciplínou a tvorbou vhodných osobných časových plánov sa dá predísť zbytočným komplikáciám. Ako si však takéto plány zostavovať? Skúsil som sa inšpirovať metódami tvorby plánov pri veľkých softvérových projektoch. Ak dobre fungujú tam, prečo by nefungovali aj pri vytváraní plánov „v malom"? Samozrejme treba tieto metódy patrične zjednodušiť. Nemá predsa zmysel stráviť pol pracovného dňa tvorbou plánu na úkor práce na pridelených úlohách. Snažil som sa však myšlienku týchto metód zachovať.

Ako postupovať pri tvorbe osobného časového plánu?

Prvým dôležitým krokom tvorby plánu softvérového projektu býva definovanie jeho rozsahu. Tu sa vymedzuje, čo do projektu patrí a čo už nie z hľadiska výstupov. Tento krok je kriticky dôležitý aj pri tvorbe osobných plánov. Avšak nie je potrebné robiť zdĺhavú a zložitú analýzu požiadaviek. Všetky potrebné informácie obsahuje dokumentácia k danej úlohe, ktorú je potrebné poriadne preštudovať. Ak ju k dispozícii nemáme, najlepšie bude obrátiť sa s otázkami rozsahu na analytika.

Bohužiaľ, to, čo všetko daná úloha obsahuje, nám iba vytvorí hrubú predstavu o jej obtiažnosti. Ťažko sa dá z týchto informácií určiť, ako dlho nám bude trvať jej implementácia. Odporúčam zamyslieť sa chvíľu nad touto úlohou a spísať si niekam v bodoch, čo všetko je potrebné spraviť pri implementácii. Pre lepšiu predstavu uvediem príklad. Povedzme, že mám za úlohu naprogramovať správu súborov. Z dokumentácie vyčítam, že úloha zahŕňa vytváranie, editovanie, mazanie a výpis súborov. Po hlbšom zamyslení však prídem na to, že budem potrebovať napríklad databázovú tabuľku, kde budem ukladať potrebné informácie o vytvorenom súbore a jeho autorovi. Ďalej budem pravdepodobne potrebovať konfiguračný súbor, odkiaľ budem načítavať základ cesty k adresáru obsahujúceho tieto súbory, aby som pri zmene cesty nemusel prepisovať kód a tiež budem potrebovať validáciu na unikátnosť názvu súboru. Tieto definované podúlohy by mali byť natoľko detailné, aby som vedel pri nich odhadnúť čo najpresnejšie dĺžku jej implementácie. No prílišná detailnosť môže byť zas zvádzajúca. Táto činnosť je analogická s metódou dekompozície, ktorú využívajú manažéri plánovania pri tvorbe plánu v softvérových projektoch. Predpokladám, že väčšina vývojárov si robí tento zoznam úloh intuitívne. Pre úplnosť je však potrebné to spomenúť.

Keď máme zoznam hotový, ostáva pred samotnou kompletizáciou plánu ešte posledný a najdôležitejší krok – odhad dĺžky trvania týchto činností. Tento krok je bohužiaľ tiež najťažší, a práve na ňom krachuje najviac plánov. Pri týchto odhadoch sa môžeme inšpirovať metódami, ktoré sa využívajú na úrovni projektov. Pre tieto účely sa už na prvý pohľad zdá byť najvhodnejšia metóda využívajúca analógiu s predchádzajúcimi úlohami. Tento spôsob však predpokladá, že mám k dispozícii informácie o mojich predchádzajúcich úlohách.

Pre tento účel navrhujem vytvoriť si evidenciu reálnych časov trvania úloh, ktoré mi boli pridelené v minulosti. Tieto údaje by bolo vhodné rozšíriť aj o krátky popis úlohy a informáciu, aké problémy bolo pri implementácii potrebné vyriešiť a čo zdržalo postup práce. Príkladom vhodnej poznámky môže napríklad byť, že z celkového času stráveného prácou na úlohe som riešil tri hodiny problém s pripojením k databáze. Túto poznámku potom môžem alebo nemusím brať do úvahy pri odhadovaní nových úloh. Odporúčam preto radšej písať podobné poznámky, ako od celkového času odpočítať čas strávený prácou, ktorá úplne nesúvisí s danou úlohou, ale bolo ju potrebné spraviť.

Trúfam si tvrdiť, že čím viac podobných záznamov bude mať vývojár k dispozícii, tým budú jeho odhady presnejšie. Pri tom sa nemusia predchádzajúce úlohy nutne podobať na tie aktuálne. Stačí si len uvedomiť, o koľko je zhruba aktuálna úloha zložitejšia ako tá z minulosti, ktorá mi trvala napríklad desať hodín a o koľko je jednoduchšia, ktorá trvala pätnásť hodín. Na základe týchto informácií dokážem približne odhadnúť čas potrebný na implementáciu novej úlohy.

Skúsenosti však postupom času narastajú a tým narastá aj efektivita a rýchlosť práce. Preto sa stávajú najstaršie záznamy irelevantné a mohli by zbytočne vnášať do odhadov nepresnosti. Navrhujem ich preto postupne mazať. Mal by však ostať vždy dostatočný počet záznamov na porovnanie.

Iný pohľad na vec predkladajú psychológovia tvrdením, že pri odhadovaní času potrebného na ukončenie úlohy sme príliš optimistickí v prípade, že tento odhad vytvárame pre seba [1]. Na druhú stranu v prípade, že odhadujeme potrebný čas pre niekoho iného, vieme reálnejšie odhadnúť prípadné problémy a komplikácie. Z tohto dôvodu sa v agilných metódach vývoja softvéru odhaduje čas potrebný na ukončenie úlohy skupinovo. Preto majú výhodu členovia SCRUM tímov, ktorých osobné časové plány sa dajú jednoducho odvodiť z výsledkov stretnutia, kde preberajú časové zložitosti úloh šprintu.

V iných metódach vývoja softvéru si však neviem predstaviť, že by sa niekoľkí kolegovia podieľali na tvorbe osobného plánu niekoho iného. Nikdy však nie je na škodu popýtať si radu od skúsenejšieho kolegu, najmä v prípade, že nemáme dostatočné skúsenosti z predchádzajúcich úloh a teda nemáme z čoho vychádzať. Povahou sa však tento spôsob podobá skôr na expertný posudok.

Optimizmus je zlý radca

Ako som už naznačil skôr, ľudia majú tendenciu príliš optimisticky pristupovať k vytváraniu vlastných osobných plánov. Dôvodom býva väčšinou predstava o tom, ako danú úlohu úspešne zvládnu a pri tom nie sú schopní reálne zhodnotiť prípadne riziká. Americký psychológovia sa vo svojej štúdii [1] zaoberali dôvodmi tohto fenoménu. Jednou z hlavných príčin označili fakt, že ľudia nezohľadňujú vlastné skúsenosti a prípadné neúspechy sa snažia vysvetliť zlyhaním externého faktoru, ktorý nemohli ovplyvniť.

Práve toto riziko optimistického odhadu potláča môj návrh evidencie ukončených úloh, ktoré som opísal v predchádzajúcej kapitole. Odhad totiž nielenže zohľadňuje, ale priamo vychádza z predchádzajúcich skúseností.

Obdobný spôsob evidencie predstavuje softvérový inžinier a spisovateľ Joel Spolsky na svojom blogu *Joel on software* [2]. Odporúča rozdelenie úloh na podúlohy, ktorých

odhadnutý čas implementácie nepresiahne 16 hodín. Spolsky navrhuje, aby si každý vývojár zaznamenával históriu odhadnutých a skutočných trvaní úloh. Na základe týchto údajov je potom možné vytvoriť graf reprezentujúci správnosť odhadov. Tvrdí, že pomocou tohto grafu je možné lepšie odhadnúť čas nových úloh.

Osobne si to však neviem v praxi predstaviť. Z grafu som schopný vyčítať akurát to, či som v odhadoch dostatočne dobrý a nakoľko môžem veriť svojmu úsudku. Mnou navrhovaná metóda je efektívnejšia, nakoľko každý vývojár hneď vidí podstatu predchádzajúcich úloh spolu s ich trvaním a môže zvážiť podobnosť s odhadovanou úlohou. Za zváženie by však stála kombinácia oboch metód a teda rozšíriť moju metódu aj o informáciu o odhadovanom čase.

Čo ak...

Dobrým zvykom býva zahrnúť do odhadov rezervy kvôli možným neočakávaným komplikáciám. Robí sa to aj pri vytváraní plánov celých projektov, ako i pri vytváraní osobných plánov.

Opisovať výhody je zbytočné, no málokto si však uvedomuje, že príliš veľké časové rezervy môžu mať negatívny dopad na samotný plán. Zas v tom hrá rolu psychológia a prirodzená ľudská pohodlnosť. Ak mám na úlohu vyhradených nasledujúcich 5 dní, prečo sa ponáhľať a nepozrieť si najprv emaily a nedať si pauzu na cigaretu? Ak si ale naplánujem čas úlohy na 2 dni, tak si rozmyslím, či pol dňa preflákam podobnými činnosťami.

Ďalším vedľajším efektom je fakt, že väčšina vývojárov natiahne čas implementácie do takej miery, aby vyplnili všetok rezervovaný čas. Preto sa pokojne môže stať, že úloha, ktorá by bola za normálnych okolností hotová za 2 dni bude trvať podstatne dlhšie. Toto je tiež vecou pracovnej morálky.

Ako si teda zvoliť vhodnú časovú rezervu? Na takúto otázku nie je jednoznačná odpoveď. Treba ju zvoliť vhodne a s ohľadom na vyššie zmienené skutočnosti. Zaujímavou možnosťou môže byť rada, ktorú som dostal od kolegu z práce. Nami odhadnutý čas úlohy vynásobíme odmocninou z troch alebo odmocninou z dvoch v závislosti od toho, či považujeme úlohu za ťažšiu, alebo ľahšiu.

Záver

Plánovanie je jednou z najdôležitejších činností spojených s vývojom softvérového projektu. Bez neho by sa asi máloktorý projekt úspešne ukončil. Plánovanie by však nemal byť cudzí pojem ani pre samotných programátorov. Tí by si mali obdobným spôsobom vytvoriť plán práce na svojich úlohách. Inšpiráciou im môžu byť práve metódy, ktoré sa využívajú na úrovni celých projektov. Nie všetky metódy sú však vhodné kvôli ich zložitosti a časovej náročnosti.

Plánovanie sa nezaobíde bez odhadu trvania činností. Samotní programátori však túto činnosť nemajú radi. Jednak z dôvodu netriviality tejto úlohy a jednak z toho dôvodu, že sa potom tohto plánu musia držať.

Navrhovaná evidencia ukončených úloh spolu s informáciou o ich trvaní by mohla podstatne uľahčiť tvorbu odhadov a tým aj urýchliť tvorbu osobných plánov. Pri ich

tvorbe by sme nemali zabúdať na fakt, že príliš veľká časová rezerva môže spôsobiť viac škody ako úžitku a tiež môže byť živnou pôdou pre prokrastináciu. Naším prvoradým cieľom je vytvoriť predsa plán, ktorý bude splniteľný a nebude nám spôsobovať časový stres.

Použitá literatúra

1. Buehler, R., Griffin, D., Ross, M.: Exploring the „Planning Fallacy“: Why People Underestimate Their Task Completion Times. Journal of Personality and Social Psychology, Vol. 67, No. 3, 1994, 366-381.
2. Spolsky, J.: Evidence Based Scheduling. 2007. Dostupné na internete: http://www.joelonsoftware.com/items/2007/10/26.html
3. Steel, P.: The Nature of Procrastination: A Meta-Analytic and Theoretical Review of Quintessential Self-Regulatory Failure. *Psychological Bulletin*, Vol. 133, No. 1 (2007) 65-94.

Annotation

Effective time planning

Planning is undoubtedly very important. I tis said that i tis better to have a bad plan, as have no plan. This applies not only for managers, but also for developers. They should plan work on theyr tasks. This essay provides a way, how to efficiently create personal plans. It also suggests a method to simplify time estimation.