Tvorba plánov v softvérovom projekte, rozdelenie úloh, plnenie a aktualizácia plánov projektu

JURAJ MAJER

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jmajer@chello.sk

Abstrakt. Cieľom tejto eseje je podať ucelený pohľad autora na problematiku plánovania v softvérovom projekte, stanovenie dôležitosti jednotlivých cieľov, plánovanie úloh pri práci na vývoji softvéru a ich rozdelenie medzi jednotlivých členov tímu. Existuje niekoľko prístupov k tvorbe plánov. V tomto dokumente sa bližšie sústredíme na porovnanie dvoch pohľadov – tradičného vodopádového modelu plánovania a špirálového prístupu k plánovaniu a riadeniu softvérového projektu. Ďalej uvedieme charakteristiky extrémneho programovania ako jedného z prístupov k rozdeleniu úloh v tíme. Plány sa vytvárajú na to, aby usmerňovali ľudí, ktorí sa snažia ich dodržiavať. Každý proces však musí byť súčasťou diskusie, postupne sa meniť a zdokonaľovať. Správny plán je organický, zahŕňa zvyky a názory tímu, reflektuje na prípadné zmeny počas trvania projektu. Preto pri dodržiavaní plánov a riadení ľudí treba dbať na možnosť ich aktualizácie. Striktné dodržiavanie obmedzení môže viesť v tíme k nepochopeniu a odmietaniu spoločného ducha a úplne znemožniť vytvorenie žiadaného synergického efektu.

Úvod

Jedným z problémov tvorby softvéru je ťažká vizualizácia postupujúceho riešenia. Je jednoduchšie sledovať pokroky v stavbe budovy ako pozorovať vývoj softvérového projektu. Často sa stáva, že neskúsení manažéri a vedúci tímov precenia svoje schopnosti a schopnosti svojich podriadených tvoriť a dizajnovať nový produkt . To potom vedie k prekročeniu konečných termínov. Prekročenia o 100 alebo 200% sú bežnou záležitosťou [1]. V roku 1995 bolo letisko v Denveri v štáte Colorado mimo prevádzky kvôli neočakávanej poruche softvéru, ktorý kontroloval batožinový systém. Celková strata letiska sa vyšplhala na 1,1 milióna dolárov denne + operačné náklady.

Na to, aby bol softvérový projekt úspešný, je potrebné vytvoriť podrobný plán činností, ktorý bude flexibilný a bude presne odrážať povahu situácie, v ktorej má byť systém vyvíjaný. Podľa našich pozorovaní u mnohých ľudí prevláda názor, že

Manažment v softvérovom inžinierstve, október 2006, s. 1-7.

manažment a tvorba plánov je potrebná, napriek tomu jej však mnohí nevenujú takú pozornosť, ako by si zaslúžila. Plán je prvým krokom pri založení nového projektu a dáva na dlhú dobu jediný obraz o tom, ako má vyzerať konečné riešenie. Projektový plán obsahuje rozvrh činností a časových medzníkov (termínov), ktoré majú vyústiť do dosiahnutia požadovaného cieľa. Je však logické, že plán je iba odhad trvania činností a ich postupností a odhad potrebného úsilia vynaloženého jednotlivými členmi tímu. Slovo odhad v predchádzajúcej vete naznačuje, že plán nie je nemenný, ale sa časom spresňuje a vyvíja. Je preto potrebné, aby bol flexibilný a aby bolo možné ho hocikedy aktualizovať. V odbornej literatúre je veľa zmienok o potrebe plánovania, nikde však nenájdeme návod ako plán projektu zostrojiť. Je to preto, lebo plánovanie úzko súvisí s povahou konkrétnej situácie a čo je ešte závažnejšie, súvisí s povahou jednotlivých členov tímu a samotného vedúceho. Proces plánovania nemôžeme charakterizovať bez zmienky o tom, že je mimoriadne dôležité, kto tento plán vytvoril a kto dohliada na jeho plnenie. V nasledujúcich riadkoch sa pokúsime čitateľovi predostrieť našu predstavu o tom, čo by mal úspešný plán obsahovať.

Plán by mal jasne definovať ciele projektu. Konečné ciele je možné rozložiť na menšie ciele, ktoré je potrebné splniť, aby dali dokopy želateľný výsledok. Na základe vytýčených cieľov sa určia činnosti, ktoré treba vykonať. Plán má určiť jednotlivým členom tímu, čo majú robiť a kedy to majú urobiť (v akej časovej postupnosti plniť úlohy). Plán musí obsahovať definíciu jednotlivých procesov-činností, a teda aj štádium, kedy sa daný proces považuje za splnený. Postoj manažéra, ktorý dbá na dodržiavanie plánu, musí byť veľmi citlivý. Pri prekročení času vyhradeného na implemetovanie daného procesu je potrebné definovať príčiny, prečo sa úloha nestihla spraviť v danej dobe a vyvodiť dôsledky. Ak sa vyskytli neočakávané skutočnosti, je potrebné plán aktualizovať. Ľudský faktor je v tomto procese kľúčový. Plán musí naviac obsahovať aj odhad vynaložených prostriedkov (ľudská práca, hardware, iné zdroje) a poskytovať ucelenú predstavu o konečnom riešení. Tieto vedomosti sú v rannom štádiu riešenia projektu ešte zahmlené a tvorca plánu musí sčasti „vidieť do budúcnosti“. Pri tejto úlohe mu pomôže diskusia s dizajnérmi, technickými poradcami a vyvojármi projektu.

Prístupy k tvorbe plánov

V tvorbe plánov musíme vyriešiť otázku, ako bude vyzerať životný cyklus softvérového projektu. S tým totiž súvisí rozvrh činností a ich časová dimenzia. V našej eseji sa budeme venovať dvom modelom plánovania životného cyklu projektu:

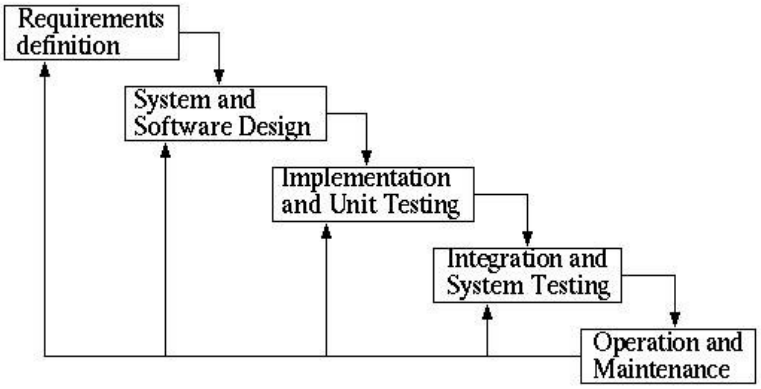
- vodopádový model (tiež známy z angličtiny ako SDLC model – System Development Life Cycle model)
- špirálový model (pre praktické skúsenosti viď [1])

Vodopádový model riadenia životného cyklu projektu obsahuje 7 jasne definovaných etáp, ktoré na seba nadväzujú a jedna nesmie predbiehať druhú.

Z pohľadu plánovania je teda presne určená časová postupnosť a otázkou zostáva stanovenie dĺžky jednotlivých etáp.

Jednotlivé etapy (viď Obr.1.):

1. *Analýza a špecifikácia požiadaviek* – plánovanie základných funkcionalít systému. Čo má výsledný produkt spĺňať.
2. *Architektonický návrh* – ujasňuje sa celková koncepcia systému.
3. *Podrobný návrh* – podrobná špecifikácia softvéru. Treba vytvoriť plán prác na implementáciu. V Obr.1. je 2. a 3. fáza spojená do jednej etapy.
4. *Implementácia a testovanie súčiastok* – programová realizácia softvéru a jeho testovanie.
5. *Integrácia a testovanie systému* – spájanie podsystému do celistvého systému.
6. *Akceptačné testovanie a inštalácia* – testovanie systému používateľom.
7. *Prevádzka a údržba* – zabezpečenie prevádzky softvéru. V Obr.1. je 6. a 7. fáza spojená do jednej etapy.



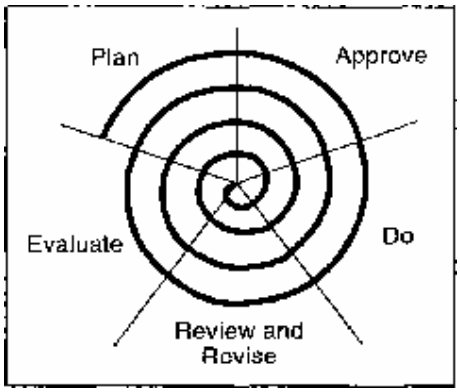
Obr. 1. Vodopádový model životného cyklu softvérového projektu [4].

Cieľom tejto eseje nie je podrobne vysvetliť štruktúru vodopádového modelu životného cyklu projektu, ale ukázať, že pri tvorbe plánov je kľúčovou otázkou práve voľba spôsobu riešenia softvérového systému. Vo vodopádovom modeli riešenia sa pri plánovaní pristupuje k problému ako k jednotnému celku. To však podľa nášho názoru nie je najoptimálnejšie riešenie, pretože nedáva dostatočný priestor na možnosti aktualizácie, resp. zmeny plánu v prípade neočakávaných skutočností. Každá chyba v určitej etape riešenia projektu sa neskôr draho vypomstí. To, že človek robí chyby, je samozrejmé a treba s možnosťou ich neskoršieho odhalenia rátať a snažiť sa vytvoriť taký model, aby odstránenie takýchto chýb bolo čo najlacnejšie. Jednoducho povedané, plán nie je flexibilný, keď počíta v prvom období s analýzou a návrhom a v ďalších

etapách už nie je týmto činnostiam vyhradený žiaden priestor. Na druhej strane je výhodou to, že na začiatku projektu sa spraví globálna analýza, špecifikácia požiadaviek a architektonický návrh, ktoré sa potom môžu stať základom pre plánovanie projektu a konečný odhad vynaložených prostriedkov [2].

Druhým prístupom je *špirálový model*. V tomto prípade sa pri plánovaní rozdelí problém pomocou analýzy zhora-nadol na niekoľko nezávislých podproblémov (z angl. stages). Vytvoria sa tímy, ktoré majú za úlohu spoločne analyzovať, čo je potrebné spraviť na dokončenie každého podproblému. Veľkosť tímov a podproblémov musí byť vhodne zvolená, aby mal tím možnosť ukončiť prácu a implementáciu jedného podproblému za 4 až 6 mesiacov. Potom nasleduje proces plánovania riešení jednotlivých podproblémov. Myšlienkou celého špirálového modelu je fakt, že menšie projekty sa ľahšie plánujú, riadia a riešia ako veľké úlohy. Plánovací proces zhora-nadol pokračuje aj v rámci každého podproblému, až je ten rozdelený na niekoľko malých častí zvaných moduly. Každý modul by mal obsahovať činnosti, ktoré je možné dokončiť jedným človekom za 2 týždne. Náplňou jednotlivých modulov môže byť vytváranie tried v objektovo-orientovanom prístupe programovania, testovanie, tvorba dokumentácie, analýza a návrh riešenia nových problémov alebo oprava chýb. Životný cyklus každého modulu sa skladá z piatich fáz (viď Obr.2.):

1. *Plánovanie* – táto fáza je splnená, ak existuje ucelený pohľad na implementáciu a testovanie modulu.
2. *Schválenie* – táto fáza je splnená, ak je plán schválený celým tímom.
3. *Implementácia* – ukončí sa, keď vývojár skončí prácu na module.
4. *Kontrola* – vývojár ukončí testovanie a kontrolu svojej práce.
5. *Hodnotenie* – vedúci tímu a ďalší členovia vyhodnotia prínos modulu a jeho ďalšie využitie v budúcnosti.



Obr.2. Špirálový model životného softvérového cyklu projektu [1].

Podľa nášho názoru spočívajú výhody špirálového modelu v jeho flexibilite. Ak sa vývoj jedného modulu spomalí, je jednoduché na vzniknutú situáciu rýchlo reagovať a zmenami v ďalších moduloch sa snažiť o jej nápravu. Je taktiež ľahké stanoviť, kde sa stav riešenia projektu nachádza alebo aký sa za posledný čas urobil pokrok v riešení. V neposlednom rade sa pri takomto podrobnom pláne (rozdelenom na moduly) dá odhadnúť množstvo vynaložených prostriedkov (ľudská práca, hardware, iné zdroje). Nevýhodou však zostáva fakt, že na to, aby sme vedeli rozdeliť problém na podproblémy a moduly, potrebujeme architektonický model, a teda sa zo začiatku nevyhneme väčšej analýze, ako to bolo v prvých fázach vodopádového modelu.

Na záver tejto časti uveďme, že voľba prístupu k tvorbe plánov v softvérovom projekte závisí od konkrétnej situácie (veľkosť tímu, finančné zdroje, množstvo času). V niektorých prípadoch je nutné oba prístupy kombinovať alebo naopak siahnuť po úplne inom. Nás však viac oslovil špirálový model riadenia softvérového projektu.

Rozdelenie úloh v tíme

Úlohy v tíme by sa dali rozdeliť do dvoch skupín – *vývojové úlohy* a *riadiace (dozorné) úlohy.* Riadiace úlohy riešia väčšinou manažéri a patrí medzi ne hlavne:

- identifikovanie cieľov
- rozdelenie úloh a členov pre vývojové úlohy
- získavanie zdrojov a odstraňovanie prekážok
- monitorovanie a vyhodnocovanie stavu projektu
- komunikácia s okolím

Vývojový tím vytvára konečný produkt a manažéri, resp. riadiaci tím by im mal počas ich práce pomáhať, viesť ich a dozerať na plnenie jednotlivých cieľov a častí plánu. Jednotlivé úlohy vo vývojovom tíme sú taktiež dosť špecifikované a závisia od konkrétneho projektu. V každom vývojárskom tíme by však nemal chýbať:

- vedúci tímu, ktorý koordinuje tím a najviac komunikuje s manažérmi a zákazníkom
- vývojár, ktorý implementuje projekt
- technický poradca, ktorý rieši technické aspekty implementácie a architektonické otázky
- archivár, dokumentarista

Dôležité však je, aby daná štruktúra a rozdelenie úloh vyplynulo prirodzene zo situácie v tíme. Je podľa nás potrebné, aby o štruktúre prideľovania úloh rozhodovali všetci členovia a takéto otázky by sa mali riešiť na spoločných stretnutiach. Nemalo by ísť o striktné nariadenie zo strany riadiaceho tímu. Taktiež štruktúra plánov, ktorá sa vytvára v rovnakom období ako rozdeľovanie úloh, by mala byť výsledkom konsenzu všetkých členov. Ako zaujímavý prístup k rozdeľovaniu úloh v tíme sa nám javí

princíp extrémneho programovania. Tu pracujú programátori a všetci členovia tímu spoločne, teda každý sa z veľkej časti dokáže orientovať v práci svojich kolegov. Nepracuje sa za zavretými dverami, ale práve naopak, dôležitými hodnotami sú jednoduchosť, komunikácia, vplyv a odvaha. Programuje sa v pároch, čo je výhoda, pretože každá časť zdrojového kódu je prezretá minimálne dvoma ľuďmi, takže sa zmenšuje pravdepodobnosť chyby a zvyšuje synergický efekt.

Kontrola plánov

Kontrola plánov by mala v tíme prebiehať na všetkých úrovniach. Tým, že každý člen tímu má svoj vlastný osobný rozvrh, môže si kedykoľvek skontrolovať, či plán dodržiava, resp. podľa osobného plánu usmerňovať svoju aktivitu. Ako sme už v úvode naznačili, plány by mali byť súčasťou väčšej diskusie, mali by sa meniť a reflektovať tak na vzniknuté situácie. Nemali by sa stať predmetom súťaženia, kto splní viac úloh v čo najkratšom čase, ale mali by napomáhať koordinovať celkové riešenie projektu.

Na ďalších úrovniach sú plány kontrolované vedúcim tímu a manažérmi. Aktualizácie plánov by sa mali robiť na spoločných stretnutiach celého tímu, kde by každý mohol povedať svoj názor a prispieť tak k vytvoreniu lepšieho a konzistentnejšieho časového rozvrhu úloh.

Pri dodržiavaní plánov a časového harmonogramu je dôležitá aj otázka motivácie pracovníkov. Ak to okolnosti vyžadujú (málo času na implementáciu produktu), je možné, aby manažéri motivovali finančne alebo inými výhodami skoré dokončenie danej etapy implementácie.

Záver

V eseji *Tvorba plánov v softvérovom projekte, rozdelenie úloh, plnenie a aktualizácia plánov projektu* sme sa zamerali na vysvetlenie pojmu plánovanie a jeho jednotlivých súčastí. Pre tvorbu plánu je nevyhnutné zvoliť si celkový prístup k životnému cyklu softvérového projektu. V dokumente sme diskutovali dva princípy – tradičný *vodopádový model* a *špirálový model*. Podľa nášho názoru je pre súčasnosť prijateľnejší špirálový model, ale nechceme tým povedať, že je jednoznačne lepší, pretože voľba závisí na konkrétnych podmienkach. Ďalej sme uviedli náš názor, ako by malo vyzerať rozdelenie úloh v tíme a ako by sa malo pristupovať k aktualizácii a zmene plánov softvérového projektu.

Je potrebné si uvedomiť, že prvý pokus, hoci aj na počiatku veľkej myšlienky, býva väčšinou chybný. Preto nie je žiadnou katastrofou vytvorené plány meniť a prispôsobovať novým podmienkam.

Použitá literatúra

1. Rettig, M., Simons, G.: A Project Planning and Development Process for Small Teams. *Communications of the ACM*, Vol. 36, No. 10 (1993) 45-55.
2. Paulish, D.J., Nord, R.L., Soni, D.: Experience with Architecture-Centered Software Project Planning. *Proc. of the 17th International Conference on Software Engineering*, Seattle, WA, 1996, p.126-129.
3. Ching-Seh, W., Simmons, D.B.: A Knowledge-Based Approach for Dynamic Software Planning and Tracking. In: *Proc. of the Twenty-Fourth Annual International Computer Software and Applications Conference (COMPSAC '00),* October 2000, pp. 305.
4. Kessler, D. 1999. Software Process. (dátum prístupu: 20.10.2006) http://www.cis.upenn.edu/~cse350/Spring99/Lec02.htm

Annotation

Software Project Planning, Tasks Planning, Fulfilling and Updating of Project Plan

This paper describes our feeling and experiences with planning in software project, defining project goal and preparing schedule of working activities for each team member in development process. There are several ways how to plan development process. We have aimed to describe two views– traditional waterfall model of software development and spiral life cycle. Then we try to present extreme programming as an new approach to distributing project activities among team members. Plan are created to guide people who try to obey them. Each process is matter of discussion. If your process isn't changing, it is probably isn't being used. A successful plan is organic, embodied in the habits and feelings of the team. One of the most important things in project planning is hence the ability to update our plans and future visions. Strict attempts to enforce plan can lead to misunderstanding and even rebellion in the team and bury synergic effect.