AKO PLÁNOVAŤ ZMENY PLÁNOV

Bol čas pripravovať sa a pripravil som sa, ako som sa pripravil, dnes je čas s vierou vykročiť vpred.

Roman Mészároš

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

Abstrakt. *Plánovanie sa vyskytuje vo všetkých fázach projektu, je dôležitým pre dohodnutie sa klienta s vypracovávateľom projektu. Jeho súčasťou je aj odhadovanie vynaloženého úsilia, zdrojov a tvorba rozvrhu. Na plánovanie existujú rôzne metódy, ktorých úspešnosť je silne závislá od povahy projektu a tímu. Táto esej sa zameriava na porovnanie rôznych prístupov k plánovaniu a odhadovaniu, rozoberá aké atribúty projektu majú najväčší vplyv na ich výber. Keďže odhady sa uskutočňujú vo všetkých fázach projektu, je tiež nevyhnutné byť pripravený na zmenu odhadov z raných štádií projektu, čím sa tiež zaoberá táto esej. Rozoberám tiež možné zmeny a riziká, ktoré vplývajú na zmenu plánov a odhadov a tiež vplyv dodržiavania plánu na celkový výsledok projektu. V závere sa venujem spojeniu analyzovaných vecí s našim tímovým projektom.*

Kľúčové slová: *plánovanie, zmena plánov, chybovosť*

Úvod

Plánovanie nasleduje okamžite po inicializácii projektu, do istej miery je aj súčasťou inicializácie. V počiatočnej fáze slúži najmä na stanovenie vzťahu so zákazníkom a tiež ako prvotná špecifikácia projektu [1]. Plánovanie prebieha najmä v počiatočných fázach projektu a obvykle netrvá dlho, predstavuje len malú časť projektu, avšak veľmi dôležitú pre celú ostatnú prácu na projekte.

Plánovanie pozostáva z rôznych častí: stanovenie rozvrhu, rozpočtu, odhady trvania jednotlivých fáz a častí projektu, odhady nákladov, stanovenie rozpočtu. Na sledovanie a vytváranie plánov a sprehľadnenie práce sa používajú najmä tieto metódy:

- Ganttova schéma [5]
- Metóda kritickej cesty (CPM) [8]
- PERT [10]

Ganttova schéma je nástroj, ktorý zobrazuje jednotlivé úlohy v tabuľke, kde v záhlaví je dátum, kedy sa má úloha vykonať a vľavo je názov úlohy a priradené osoby na danú úlohu. V tejte schéme je tiež možné prehľadne zobrazovať, ktoré úlohy musia byť splnené pre možné pokračovanie alebo začatie ďalších, teda ich následnosť.

Pri metóde kritickej cesty (CPM) potrebujeme opäť jednotlivé úlohy a ich trvanie a vzájomnú závislosť. Následne CPM vypočíta najdlhšiu cestu naplánovaných úloh až k ukončeniu projektu a tiež najdlhšiu a najkratšiu dobu trvania jednotlivých úloh tak, aby sa nepredĺžil termín projektu. Na základe tohto oddelíme kritické úlohy, ktorých nedodržanie má silný vplyv na posun termínu projektu, a voľnejšie úlohy.

Metóda PERT sa obvykle používa v spojení s metódou kritickej cesty. Metóda PERT používa graf na zobrazenie aktivít ako uzly a vzťahy, následnosti medzi aktivitami ako hrany medzi nimi. V tomto grafe nájdeme kritickú cestu, na ktorej nám úlohy určujú dobu trvania projektu.

Nástroje na plánovanie projektu a sledovanie úloh

Najpoužívanejším nástrojom na plánovanie sa stal Microsoft Project. Tento umožňuje tvorbu plánov, priraďovanie zdrojov k úlohám, sledovanie stavu úloh, správu rozpočtu a tiež analýzu zaťaženia jednotlivých členov tímu. Je schopný vypočítať kritické plány, Ganttove schémy a obsahuje podporu pre rôzne skupiny používateľov. Je to robustné riešenie a vyžaduje samostatnú inštaláciu u každého používateľa.

Ďalším vhodným nástrojom na sledovanie úloh a pracovníkov na projekte je Jira. Pomocou nástroja Jira je možné zadávať požiadavky klientom, úlohy si jednotliví účastníci posúvajú a po každej ukončenej fáze úlohy sa táto môže uzavrieť a poslať do ďalšej fázy, z ktorej sa v prípade problému môže opäť vrátiť späť. V nástroji Jira je tiež možné k úlohám pridávať prílohy, verzie, vytvárať projektovú hierarchiu úloh a komentáre [11]. Na úlohe môžu v rôznych rolách byť zúčastnení aj viacerí používatelia súčasne, rovnako ako MS Project umožňuje tvorbu prehľadných grafov sledovanie presnosti odhadov, vkladanie príloh a pracuje ako serverová aplikácia s prihlasovaním prostredníctvom webového prehliadača.

V prípade začatia práce na projekte je vhodnejšie použitie MS Project, keďže je prehľadnejší pri manažmente zdrojov a rozloženie projektu. Pri práci na už bežiacom projekte je vhodnejšie použiť systém Jira, ktorý je tiež vhodný na zavedenie po určitom behu projektu. Tento systém umožňuje flexibilnú spoluprácu so zákazníkom, a tiež aj vo firme od analýzy cez vývoj až po testovanie.

Pre prácu v menšom tíme sa však ako vhodné ukazujú aj rôzne open source riešenia, ktoré viac, či menej poskytujú základnú funkcionalitu od reportovania chýb, sledovania stavu úloh, projektov a ich rozbíjanie na drobnejšie úlohy. V našom tíme použijeme Dot

Project, ktorý má jednoduchú údržbu a nasadenie vďaka použitej kombinácii PHP s MySQL.

Riziká pri plánovaní

Štandardné nástroje na plánovanie a odhadovanie väčšinou slúžia len na zaznamenanie aktivít a k nim priradených zdrojov. Pri odhadovaní je však potrebné uvažovať aj nad faktormi, ktoré spôsobujú odklon od pôvodného plánu, či už natiahnutím projektu alebo nárastom rozpočtu [3]. Týmito faktormi sú obvykle natiahnutie jednotlivých činností, prípadne nedostupnosť vhodných zdrojov.

K týmto problémom dochádza najmä preto, lebo ľudia, ktorí daný softvérový výrobok predávajú, majú obvykle príliš optimistické plány, čo je následne veľkým problémom pre ľudí, ktorí sú za výrobu daného produktu zodpovední, keďže sa musia vojsť do tohto optimistického rozpočtu a termínu.

Vzhľadom na povahu nášho projektu, je úplne jasný termín, a tým je zbieranie požiadaviek na tvorbu rozvrhu. Keďže pracujeme na už rozbehnutom projekte a naše rozpočtové ohraničenie je náš čas na tímovom projekte, musíme si zvoliť kľúčové funkcionality, ktoré v danom systéme do najbližšieho zberu rozvrhu chceme mať, a tiež musíme zistiť počas zberu požiadaviek na rozvrh, aké sú ďalšie nároky kladené na náš systém zo strany klienta, fakulty.

Štruktúra tímu

Vzhľadom na to, že v projekte sme zapojení 7 ľudia, pričom každý z nás má isté špecifické schopnosti, v ktorých vyniká, bolo by obrovskou škodou toto nevyužiť. Keďže predmetom našej práce bude len jeden projekt budeme vychádzať z funkcionálnej topológie. Máme určených jednotlivých vedúcich zodpovedných za dokumentáciu, podporné prostriedky, vývoj, plánovanie a vedúceho projektu. Títo vedúci majú nasledovne k dispozícii zvyšok tímu, v ktorom komunikujú úlohy jednotlivým členom, sledujú plnenie úloh a tiež komunikujú medzi sebou za účelom optimálneho využitia zdrojov.

Počiatočné plánovanie

Počiatočný plán sa vytvára ešte pred uzavretím zmluvy s klientom, predmetom tohto plánovania je zistiť, či daný produkt sme schopní vytvoriť, čo nás to bude stáť, porovnať ho prípadne s inými ponukami. Na vypracovanie počiatočného plánu potrebujeme opis tohto produktu, na porovnanie potrebujeme vytvoriť strategický plán tvorby softvérového produktu a následne stanovenie kritérií na výber produktu, ktorými môžu byť napr. zisk alebo prestíž [1].

Po prijatí projektu je potrebné vytvoriť plán na zabezpečenie chodu projektu. Tento musí obsahovať najmä požiadavky týkajúce sa zdrojov, množstva a kvality práce. Všetky zložky, ktorých sa plán týka by mali byť oboznámené so svojou zodpovednosťou, plán by teda mal byť taký podrobný ako je potrebné a nie ako je možné, a tiež musí byť odsúhlasený každou kritickou zložkou zapojenou v projekte

Plány sa vytvárajú pre rôzne oblasti projektu, nevyhnutnými plánmi, ktoré by mal mať každý projekt sú:

- plán rozvrhu
- plán nákladov.

Plán rozvrhu má za úlohu určiť, aké činnosti musia byť vykonané a zdokumentovať ich, zoradiť ich, vytvoriť odhad ich trvania a na koniec vytvoriť rozvrh pomocou kompozície týchto rozdrobených úloh.

Plán nákladov určuje potrebné zdroje na vykonanie projektu začínajúc pracovnou silou, aj zariadenia a rôzne materiály, ktoré sú pri tvorbe produktu potrebné. Následne je potrebné tieto zdroje ohodnotiť, odhadnúť náklady a vytvoriť celkový rozpočet.

Ďalšími plánmi, ktoré sa zvyknú vytvárať, ale nie je to pravidlom sú:

- plán integrácie projektu
- plán rozsahu projektu
- plán zabezpečenia akosti
- plán ľudských zdrojov
- plán komunikácie
- plán rizík [1].

Odhadovanie

Odhadom sa snažíme zistiť najpravdepodobnejšie správanie , nejakú hodnotu, či už čas, prípadne potrebné zdroje alebo iné. Poznáme rôzne druhy odhadov, ktoré sa navzájom dopĺňajú a kombinujú. Analogické odhady sa snažia využiť analógiu, podobnosť s už existujúcimi, prípadne aj ukončenými projektami a použiť znalosti nadobudnuté na nich v ďalšom projekte. Empirické odhady majú za cieľ prostredníctvom analýzy údajov určiť závislosti a vzťahy medzi rôznymi charakteristikami. Teoretické modely vytvárajú numerický model, ktorý sa overuje spravidla praxou. Heuristické modely sú modely používajúce expertné odhady.

Proces odhadovania môže prebiehať rôznymi spôsobmi, a to rozbitím projektu na drobnejšie časti, tzv. dekompozícia, modelovaním softvéru, kde je cieľom zistiť, ktoré vplyvy a ako veľmi pôsobia na trvanie jednotlivých procesov, prípadne posúdením experta, kde experti dostanú požiadavky na softvér, diskutujú o ňom, vytvoria vlastné odhady a následne ich opäť v skupine diskutujú, a tento postup opakujú až kým nepríde k zhode.

Ako predchádzať zmene plánov

Vďaka použitiu nástrojov na manažment úloh máme k dispozícii prehľad o vyťaženiach jednotlivých členov tímu. Tiež vieme, kde sa jednotlivé úlohy nachádzajú – v akom sú stave, vieme zistiť efektívnosť jednotlivých ľudí, nakoľko sú schopní plány dodržiavať, prípadne o koľko rýchlejšie sú schopní pracovať ako sme naplánovali, prípadne koľko za plánmi zaostávajú. Na základe týchto údajov sme schopní upresňovať naše odhady, a tým

prispieť k spokojnosti ako na našej strane, tak aj na strane zadávateľa projektu. Tieto získané údaje vieme do projektu zapojiť dvomi spôsobmi:

- plánovaním rizík
- pravidelnými zmenami plánu.

Pri plánovaní rizík uvažujeme množstvo rizík, ktoré môžu vplývať na dĺžku projektu, pričom sa budeme snažiť o čo najlepšie využitie zdrojov tak, aby sme minimalizovali cenu projektu k dátumu ukončenia projektu.

Plánovanie s pravidelnými zmenami plánu umožňuje opravu plánov, dobiehanie omeškaní, tiež minimalizáciu rizík, ak je počiatočná analýza rizík rozsiahla.

Vzhľadom na veľkosť projektu môže réžia spôsobená pravidelnými zmenami plánu spôsobiť väčšie výdavky, ako tie, ktoré vznikajú jej neaplikovaním, pri rozsiahlejších projektoch je však nutná a tvorí základný stavebný kameň agilného programovania.

Plnenie plánov

Sledovanie plnenia plánov je u softvérových projektov veľmi špecifické, z dôvodu neviditeľnosti softvéru, kedy je častokrát obtiažne vidieť skutočný stav projektu, resp. koľko práce nám ešte na projekte zostáva. Tento problém sa zvykne nazývať aj syndróm 90% hotovo, kde máme pocit, že väčšina práce je už za nami, avšak konečné úpravy, dolaďovanie aplikácie a komunikácia so zákazníkom pred odovzdaním môže byť ďaleko viac ako 10%, ktoré sa nám na počiatku javili.

Cieľom projektu nie je splniť plán, avšak motiváciou k tvorbe plánu je čo najpresnejšie odhadnúť potrebné zdroje a čas na výrobu produktu. Pre plnenie plánov a úspešné ukončenie projektu, je nutné sledovať, čo sa z plánu už vykonalo, aká časť projektového plánu je splnená, koľko ešte zostáva a na základe miery dosahovania plánov môže byť na mieste prípadná zmena plánu.

V našom tímovom projekte budú úlohy obvykle naplánované na jeden alebo viac týždňov v rámci plánu stretnutí k projektu, kde bude následne možné vyhodnotiť ďalšiu prácu na najbližšie obdobie. Plány úloh, ktoré sú na sebe navzájom závislé a sú pridelené viacerým riešiteľom, si budú riešitelia navzájom sledovať.

Zmena plánov

V tejto časti sa budem zaoberať samotnou zmenou plánu. Na zmenu plánu vplýva najmä plnenie plánu, zmeny sa uskutočňujú. Táto zmena sa obvykle týka preusporiadania zdrojov, prípadne natiahnutia termínu alebo navýšenia rozpočtu. Zmeny môžu byť tiež vyvolané inými vplyvmi, môže to byť vytváranie ďalších modulov, možnosti zlepšenia softvéru podľa nejakej charakteristiky, zmena legislatívy alebo zmena požiadaviek zadávateľa (klienta). Do procesu celkového riadenia zmien vstupuje projektový plán, správy o výkone a plnení plánov a požiadavky na zmeny, výstupom celkového riadenia zmien sú zmeny plánu a opravné akcie [1].

Vo vývoji môže dôjsť k chybám na nasledovných miestach:

- chyby v zmenenej entite – ak bola entita nedávno zmenená, má sklon vyvolať chybu
- chyba v novej entite – entita nedávno pridaná ma tendenciu spôsobiť chybu
- dočasné chyby – entita spôsobujúca chybu, čoskoro spôsobí ďalšie chyby
- oblastné chyby – ak nejaká entita spôsobuje chybu, blízke entity majú tiež tendenciu spôsobovať chyby [9].

Podľa [6] nevznikajú však v softvéri chyby len dôsledkom množstva programového kódu, či už na základe množstva riadkov alebo modulov alebo celkovo štruktúry. Najväčšie množstvo chýb tiež nie je závislé od množstva rôznych vývojárov, ktorí prišli s daným programovým kódom do styku a nie je závislé ani od sily väzby na ostatné moduly softvérového výrobku. Najčastejším zdrojom chýb sú práve úpravy programu a ako vhodná metrika sa javí metóda priemerného veku kódu v moduloch, pričom kód starne svojou úpravou a viackrát upravovaný programový kód má oveľa nižšiu kvalitu, ako pôvodný zdrojový kód, prípadne zdrojový kód, ktorý by vznikol, keby boli všetky požiadavky už na začiatku známe.

Z hľadiska kvality zdrojového kódu a množstva chýb v ňom je vhodné mať čo najmenej úprav, avšak vzhľadom na cenu, ktorú by stálo vytváranie nového miesto upravovania pôvodného, je vhodná pravidelná refaktorizácia jednotlivých častí a zriedkavo aj celkov. V našom tímovom projekte pracujeme s existujúcim systémom, ktorý v sebe obsahuje už veľké množstvo funkcionality a bolo by zrejme takmer nemožné pustiť sa do projektu od začiatku, našou úlohou bude aj vybrať správne časti, ktoré budeme používať my. Vzhľadom na množstvo úprav hlavne v dátovom modeli je však veľmi vhodné pouvažovať nad jeho refaktorizáciou v spojení s existujúcimi ale aj novými požiadavkami na systém.

Vznik chýb pri vývoji softvéru môžeme predpokladať ako množstvo chýb, ktoré v softvéri vznikne, alebo sa môžeme snažiť o identifikáciu modulov, kde chyba vznikne [7]. Na základe identifikovania modulov, kde vznikajú chyby, alebo pri nájdení podsystémov s chybami, ktoré pri testovaní vznikli, vieme vytvárať dlhodobé plány. Tiež máme možnosť prealokovať zdroje a následne ich po čase vieme overovať, samozrejme táto metóda sa dá použiť aj pri krátkodobom plánovaní.

Existuje množstvo spôsobov, ako prideľovať správnych ľudí (zdroje) na jednotlivé projekty, úlohy, prípadne vylepšenia a riešenia chýb. V [3] sa autori zamerali na podporu zmien požiadaviek vo vývoji s otvoreným zdrojovým kódom. Ich myšlienkou bolo použiť softvérové a verziovacie úložisko ako novú príležitosť pre obrovský zdroj dát na analýzu, možnosť sledovania chýb, zmien, znovupoužiteľnosti a zložitosti softvéru. Úlohou bolo vytvoriť podporu pre projektových manažérov, pomocou ktorej môžu pre danú úlohu vybrať najvhodnejšieho riešiteľa, tiež umožniť klasifikáciu vývojárov v rámci ich zodpovedností. Najlepším riešiteľom je teda pravdepodobne ten, kto riešil podobné zmeny. Túto podobnosť vyhodnocovali na základe podobnosti komentárov a opisov k úlohám v prirodzenom jazyku, ktoré sa nachádzali v dokumentáciách alebo opisoch požadovaných zmien. V rámci tohto riešenia boli vývojári podľa ich dokumentov reprezentovaní v databáze, kde požiadavky na zmeny tvorili dopyt pre túto databázu.

Ďalšou skvelou možnosťou je vytvoriť zoznam podsystémov, ktoré budú s najvyššou pravdepodobnosťou obsahovať chyby, prípadne sa v nich chyby neskôr vyskytnú [8].

Tento zoznam sa následne predloží manažérovi, ktorý vďaka nemu vie presnejšie prideliť zdroje tam, kde budú lepšie využité. Tento zoznam vzniká na základe rôznych heuristík:

- podľa počtu predchádzajúcich chýb
- podľa množstva vývojárov zapojených do podsystému
- podľa toho, kedy bola posledná chyba v podsystéme alebo úprava
- podľa množstva úprav
- kombináciou uvedených kritérií.

Zoznam podľa najčastejšie upravovaných podsystémov vychádza z faktu, že pri množstve úprav sa stráca v zdrojovom kóde jeho pôvodná organizácia, nie je už taký čistý, ako keď bol pôvodne napísaný. Tento prístup však môže spôsobiť pri práci na nejakej časti to, že zvýšeným úsilím a prácou na nej odstránime z tohto zoznamu ostatné podsystémy, ktoré by sa tam tiež potrebovali dostať. Tento problém nazývame znečistením zoznamu [8]. Táto heuristika sa často kombinuje s výberom podľa posledne modifikovaného podsystému.

Heuristika výberu posledne modifikovaného vychádza z toho, že práve tieto zmeny môžu spôsobiť chyby, keďže sú len nedávno nasadené, je možné, že ich dôsledky sa ešte neprejavili, prípadne na práve bežiacej verzii sú kompatibilné, avšak nemusia sa správať voči zmenám tak, ako sa pôvodne uvažovalo.

Heuristika výberu najčastejšie meneného podsystému vychádza z predpokladu, že podsystémy, ktoré boli často menené a spôsobovali chyby ich zrejme budú pri nejakých príležitostiach spôsobovať opäť. Táto heuristika môže tiež spôsobiť znečistenie zoznamu.

Heuristika posledne opravovanej časti predpokladá, že časti spôsobujúce posledné chyby budú mať najsilnejšiu tendenciu spôsobovať chyby aj v budúcnosti, až kým sa väčšina chýb neidentifikuje a neopraví.

Záver

Už na začiatku práce na projekte sme si stanovili ciele, ktoré budú vyžadovať značnú zmenu existujúceho systému, či už v rámci rozhraní samotných, alebo aj prebudovania jadra. Samotná architektúra systému bude teda veľmi podobná už existujúcemu rozvrhovému systému. V prípade rapídnej zmeny bude prvotný návrh prebiehať zhora nadol a na základe toho sa bude plánovať práca toho-ktorého člena tímu, po úvodnej analýze prídu na rad črty (ang. features), ktorými by sme chceli zaujať. Vo veci plánovania rizík sa chceme rizikám v prvom rade vyhnúť, na ich riešenie nám pomôže systém plánovač úloh, podrobné zoznámenie sa s technológiami, ktoré budeme používať, porovnávanie s existujúcim systémom, prípadne komunikácia s minuloročným tímom. Keďže pracujeme s existujúcim už rozbehnutým systémom, naše pôvodné plány a odhady môžu byť dosť nepresné, budeme však fungovať so zoznamom používajúcom najmä heuristiku najproblematickejších miest.

Vzhľadom na skúsenosti minuloročných riešiteľov projektu rozvrhov očakávame aj spätnú väzbu zo strany používateľov systému. V rámci tímu už poznáme isté kvality jednotlivých jeho členov. Po začatí podrobnejšej práce na projekte sa naše kvality viac vyhrania a budeme vedieť presnejšie a zodpovednejšie priraďovať riešiteľov práve k tým úlohám, kde budú môcť pracovať, čo najefektívnejšie.

Použitá literatúra

1. Bieliková, M.: *Softvérové inžinierstvo: Princípy a manažment*. STU, Bratislava, 2000.
2. Boehm, B. et al.: *Bayesian Analysis of Empirical Software Engineering Cost Models*. In: IEEE Transactions on Software Engineering, IEEE Press, California, San Jose, 1999.
3. Canfora, G, Cerulo, L: *Supporting Change Request Assignment in Open Source Development*. In: Proceedings of the 2006 ACM symposium on Applied computing, ACM Press, New York, New York, 2006.
4. Deleris, A et al.: *Simulation of Adaptive Project Management Analytics*. In: Proceedings of the 2007 Winter Simulation Conference, IEEE Press, New Jersey, Piscataway, 2007.
5. Gantt, H.L.: *Work, Wages and Profit.* In: The Engineering Magazine, Hive Publishing Company, Pennsylvania, Easton, 1919.
6. Graves, T.: *Predicting Fault Incidence Using Software Change History*. In: IEEE Transactions on Software Engineering. IEEE Computer Society, New Mexico, Los Alamos, 2000.
7. Hassan, A.E.: The Top Ten List: dynamic fault prediction. In: Proceedings of the 21st IEEE International Conference on Software Maintenance, IEEE Computer Society, Canada, Ontario, 2005.
8. Kelley. J.E., Walker, M.R.: *Critical-path planning and scheduling.* In: Proceedings of the Eastern Joint Computer Conference, ACM, Massachusetts, Boston (1959), 160 – 173.
9. Kim, S. et al.: *Predicting Faults from Cached History*. In: Proceedings of the 29th international conference on Software Engineering. IEEE Computer Society, DC, Washington, 2007.
10. Milosevic, D.Z.: *Project Management ToolBox: Tools and Techniques for the Practicing Project Manager.* Colorado, Wiley, 2003.
11. Weiss, C., Premraj, R., Zimmermann, T., Zeller, A.: *How Long Will It Take to Fix This Bug?* In: Mining Software Repositories, ICSE Workshops MSR, Minnesota, Minneapolis, 2007.

Annotation

Plan changing plans

Planning is a very important part of client – developer communication and it takes part in every part of a project's lifecycle. It contains estimation of project effort, resources and schedule. There are plenty of different planning methods and their success is highly dependent on project type and the project team. This essay focuses on comparison of various planning and estimation strategies based on different project characteristics that have the greatest impact on them. Because estimates change all over the project life cycle, it is necessary to be prepared for changing the early estimates. This essay describes various changes and risks, which result in changing of plans and estimations and following the plans schedule on the project's result. The last part of this essay is devoted to application of the whole analysis to our team project.