KEĎ TO NEJDE, TAK TO PÔJDE

Kľudne si požicaj... ale nezabudni vrátiť.

Martin Konôpka

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
konopka09[zavináč]student[.]fiit[.]stuba[.]sk

Abstrakt. *Vytvorenie a sledovanie plánu projektu je dôležitou súčasťou plánovania. Naplánované činnosti pre dokončenie projektu však často trvajú dlhšie ako bolo predpokladané. K dispozícii máme viacero prístupov vývoja softvérového projektu. Každý disponuje iným spôsobom plánovania, čo je spôsobené hľadaním toho správneho. Softvérové projekty sú špecifické častou zmenou požiadaviek zákazníka a plánu projektu ešte počas jeho vytvárania. Manažér plánovania musí byť schopný reagovať na vzniknuté zmeny v požiadavkách alebo problémy v plnení plánu. Každý prístup ponúka rozličné techniky pre znázornenie plánu. Keďže nepoznáme recept na dokonalý plán, nemôžu byť ani tieto techniky dokonalé. No vieme spájať výhody viacerých techník. V eseji sa zaoberáme kombinovaním existujúcich techník znázornenia plánu v tradičnom a agilnom prístupe vývoja projektu s cieľom ich vylepšenia.*

Kľúčové slová: *plánovanie, tradičný prístup, vodopádový model, Gantt schéma, kritická cesta, agilný prístup.*

Bežná realita

Na úvod si pripomeňme bežnú realitu vývoja projektu. Samotné dokončenie projektu pozostáva zo splnenia rozličných činností, ktorých postup určuje plánovanie. Manažér projektu zadefinuje činnosti, zoradí ich podľa dôležitosti, vyhodnotí ich dĺžky. Výsledný plán určuje nielen rozvrh prác, ale i časové obdobie ukončenia projektu. Počas práce na projekte manažér sleduje plnenie plánu, no časom sleduje ako sa všetko začne vymykať spod kontroly. Manažér prehodnocuje svoje pôvodné odhady, upravuje plán, prípadne tlačí na pracovníkov. Nájsť riešenie ako v tejto situácii splniť pôvodný plán je veľmi

náročné. Zákazník čakal doručenie výsledku včas, je nespokojný, nerozumie prečo sa práca nestihla. Aký bol vlastne problém? Zlý plán.

Plánovanie a jeho význam

Pod vytvorením plánu rozumieme odhalenie aktivít, ktoré sú potrebné pre dosiahnutie cieľa, ich náročnosť a usporiadanie ich plnenia v čase. Cieľom plánu je dokončenie projektu, uspokojenie potrieb zákazníka i splnenie jeho požiadaviek.

Aj samotný zákazník má vďaka plánu možnosť odkontrolovať si plnenie jeho požiadaviek a či sú jeho prostriedky správne vynaložené. Manažér komunikuje so zákazníkom postup prác i aktuálny stav v plnení jeho požiadaviek, prípadne ich konzultuje ako by ich bolo možné ďalej plniť. Konzultuje ich však preto, pretože očakáva, že ich zákazník môže zmeniť? Veď po uzavretí zmluvy, podľa ktorej má tím dodať požadovaný produkt by sa už podmienky meniť nemali, inak to ohrozí dokončenie [2]. Ak si je toho zákazník vedomý, všetko je v poriadku. Prezradím však, že zákazník sa s tým v mnohých prípadoch nestotožní. Očakáva dodanie produktu v stanovenom termíne, no tie sa kvôli zmenám jeho vlastných požiadaviek posunú. Zákazník by si preto mal byť vedomý svojich rozhodnutí a možných dopadov zmien požiadaviek, na ktoré ho upozorní manažér.

Plánovanie projektu závisí od požiadaviek zákazníka a cieľov, ktoré chce mať splnené. Sú to práve softvérové projekty, ktoré sú týmto špecifické od iných projektov. Sám zákazník už pri zadávaní projektu totiž nevie čo chce. My sa snažíme identifikovať jeho ciele, skonkretizovať jeho myšlienku. Najviac sa mi pri mojich projektoch osvedčilo snažiť sa komunikovať detaily so zákazníkom hneď zo začiatku. Vidím za prínosné vžiť sa do jeho situácie a odhaliť akú má predstavu nad projektom. Zákazník má víziu o tom, čo by chcel byť schopný s výsledným produktom spraviť, v čom by mu mal produkt pomôcť. Odhalíme jeho vytýčený cieľ a počas samotného vývoja sa už len snažíme priblížiť čoraz viac a viac k spresneniu jeho požiadaviek tak, aby sme nakoniec tento cieľ splnili. Tým, že splníme zákazníkove požiadavky a on dostane čo žiadal, tak dospejeme k výhre na oboch stranách.

Spôsob plánovania projektu by mal byť kvôli týmto skutočnostiam schopný zvládnuť časté zmeny požiadaviek či smerovania projektu na najbližšie obdobie a predpokladať vykonanie častých zmien v pláne. Spôsob ako budeme plánovať projekt závisí od zvoleného prístupu vývoja projektu. Na výber máme nasledujúce možnosti:

- tradičný prístup založený na vodopádovom modeli,
- agilný prístup.

Spôsoby plánovania a ich techniky sa pre každý prístup odlišujú, pričom každá z nich má svoje výhody i nevýhody. Prečo by sme sa však mali za každú cenu vybrať len jednou cestou? Nie je možné ich kombinovať? Najprv sa vyjadrím k možnostiam striktnej voľby prístupu plánovania. Je však očividné, že výhody z jednotlivých prístupov možno skombinovať.

Voľba tradičného prístupu

Tradičný prístup využívajúci vodopádový model vývoja softvéru nie je vôbec pripravený na možné zmeny požiadaviek zákazníka. Vykoná sa zber požiadaviek, analyzujú sa možnosti riešenia, prebehne návrh a vývoj projektu. Výsledný produkt sa odovzdá zákazníkovi, ktorý dostane to, čo žiadal. V skutočnosti to skôr znamená, že dostane produkt podľa nášho pochopenia jeho predstáv. A tak môže byť výsledok niečo úplne iné ako si zákazník pôvodne predstavoval. Pracoval som na niekoľkých projektoch, kde som uplatnil tradičný prístup. Boli to však projekty malého rozsahu, kde sa nedostatky tohto prístupu neprejavili až v takej miere, akoby sa mohli prejaviť na veľkých projektoch. No je zrejmé, že je to veľký problém tohto prístupu.

To neznamená, že tradičný prístup je nepoužiteľný. Jednoduchý príklad jeho vhodnosti uplatnenia je z vývoja navzájom podobných produktov. Môžem mať skúsenosť s vývojom rady softvérových produktov a môžem dostať požiadavku od zákazníka na podobný produkt, ktorý som už v minulosti vytvoril. V takom prípade mi príde tradičný prístup ako vhodná voľba. Analýza požiadaviek je priamočiara, viem čo je možné vytvoriť, ako dlho to bude trvať a je to možné skonzultovať so zákazníkom okamžite.

Pri tradičnom prístupe vývoja vytvára manažér plán po získaní požiadaviek a vytvorení si obrazu o projekte. Stanoví dlhodobé ciele projektu a identifikuje úlohy nutné na ich splnenie. A práve najväčšou slabinou tradičného prístupu je netriviálnosť odhadu trvania a náročnosti aktivít pre splnenie týchto úloh. Z vlastných skúseností uvediem príklad, že je niekedy náročné odhadnúť ako dlho bude trvať naprogramovanie funkcionality. Môže sa zdať, že to bude pár hodín, no skončí to na dvoch dňoch z dôvodu zistenia problému v možnostiach použitého programovacieho jazyka. Ako potom odhadnúť trvanie všetkých činností na projekte? Ak ich už odhadnem, môžem predpokladať, že skutočnosť bude nakoniec iná. Navyše, prípadná zmena požiadaviek zákazníka mi celý odhad rozbije a moja práca bude zbytočná. Prečo som už skôr nepredpokladal, že sa plán prestane plniť? Môžu za to neskúsení vývojári, keď hrozia takéto zdržania? Môžu, ale len čiastočne. Najväčší podiel má na tom podľa mňa najmä závislosť na stanovovaní dĺžky trvania činností v pláne.

Identifikované úlohy a ich trvanie však potrebujem znázorniť. Pri tradičnom prístupe vývoja možno uplatniť matematické techniky [3], akými je Gantt schéma a kritická cesta. Ich najväčšou nevýhodou je však ťažké adaptovanie sa na zmeny požiadaviek zákazníkom, čo vyplýva z ich príslušnosti k tradičnému prístupu vývoja.

Gantt schéma

Gantt schéma znázorňuje úlohy na časovej osi v horizontálnom smere. Ako manažér plánovania mám z nej prehľad o úlohách, ktoré som stanovil pre splnenie jednotlivých cieľov projektu. V schéme vidím trvanie úloh, kedy musia byť splnené, ale nie ich závislosti medzi sebou. Napriek tomu, že schéma je pekná, farebná a príjemne sa na ňu pozerá manažérovi aj zákazníkovi [5], tak pri veľkom projekte presahuje niekoľko obrazoviek monitora počítača a vďaka malej hustote informácií je neprehľadná. Zmenou požiadaviek a priorít môže byť potrebné prepracovať celú schému. V tom prípade zahodím pôvodne vynaložený čas na jej vytvorenie, ktorý nie je zanedbateľný. Gantt schéma je vhodná na znázornenie plánu, ale nie jeho tvorbu.

Technika kritickej cesty

Poradie vykonania úloh môžem taktiež vyjadriť sieťou, orientovaným grafom, v ktorom prepojenia medzi vrcholmi budú úlohy a hodnoty hrán budú dĺžky ich vykonávania. Technika kritickej cesty je založená na nájdení najdlhšej cesty v sieti, ktorej dĺžka určuje trvanie projektu. Ak sa oneskorí vykonanie ktorejkoľvek úlohy na tejto ceste, oneskorí sa aj samotný projekt. Manažér tak vie, ktoré úlohy má pozorne sledovať a reagovať na problémy v ich plnení aby sa projekt splnil v čase.

Na tejto metóde oceňujem znázornenie závislostí úloh a možnosť určiť, ktoré úlohy môže tím vykonávať paralelne. Čo to znamená adaptovať graf na zmeny v pláne? Na rozdiel od Gantt schémy je to možné jednoduchšie. Preskupím vrcholy, zmením prepojenia, vykonám len lokálne zmeny.

Podobne však ako pri Gantt schéme, pri veľkom množstve úloh je sieť značne neprehľadná, narastá na objeme a závislosti úloh sa začínajú prepletať.

A čo tak agilný prístup?

Agilný prístup sa na vývoj softvéru pozerá opačne ako tradičný. Plánovanie projektu už nie je predpovedanie ďalekej budúcnosti [4]. Vývoj sa neriadi jedným termínom dokončenia projektu, ale pravidelnými iteráciami, v ktorých sa splnia v tom čase najdôležitejšie úlohy. Tím vývojárov, alebo ich zástupcovia, sa pravidelne stretávajú so zákazníkom, s ktorým konzultujú jeho ciele s predpokladom, že jeho záujmy sa môžu kedykoľvek zmeniť. Oceňujem zapojenie zákazníka do vývoja, pretože vidí postupný priebeh prác a v pravidelných intervaloch dostáva verzie produktu. Vie si sám vyhodnotiť ako je spokojný a koľko prostriedkov vyhradí pre ďalšiu iteráciu.

Na vývoji niektorých projektov, na ktorých som sa podieľal, sme uplatnili agilný prístup, konkrétne jeho metodiku SCRUM. Tá usporadúva identifikované úlohy do zoznamu usporiadaného podľa priorít zákazníka. Na začiatku iterácie (v metodike označenej termínom „šprint“) spolu s tímom vývojárov odhadnem náročnosť úloh a podľa skúseností z predchádzajúcich iterácií vyberiem tie úlohy, ktoré naplnia používateľove požiadavky a tím ich dokáže zvládnuť. Tým pádom sa úlohy plánujú len na najbližšiu iteráciu, a tak nemožno plánovať niekoľko iterácií dopredu.

Ak je iterácia primerane krátka, tak techniky Gantt schéma a kritická cesta sa zdajú byť zbytočnými. S tým sa však nestotožňujem. Znázornenie úloh zoznamom v metodike SCRUM je príliš jednoduché a menej prehľadné pre odvodenie závislostí medzi úlohami. Preto vidím priestor pre využitie výhod techník z tradičného prístupu.

Kombinovanie techník

Každá technika má pri svojom druhu vývoja softvéru svoje nedostatky. Ak nám v nami vybranej technike chýbajú výhody inej, prečo si ich nepožičať? Navrhujem pokúsiť sa obohatiť plánovanie oboch spomenutých prístupov vývoja projektu.

Prepojenie úloh v Gantt schéme

Uvažujme malý projekt, požiadavky zákazníka máme dostatočne odkomunikované a neočakávame veľké zmeny v zadaní. Preto sa rozhodnem o tradičný spôsob manažovania projektu a použitie Gantt schémy na zachytenie nášho plánu. Identifikujem všetky potrebné úlohy, no zo schémy neviem identifikovať, ktoré úlohy možno vykonať paralelne. Ak by som mal nájsť riešenie, hľadal by som vzájomné závislosti úloh a podľa toho ich usporiadal v Gantt schéme. Z Gantt schémy viem potom vyčítať, ktoré úlohy sú navzájom závislé.

Gantt schéma v agilnom prístupe

Nezastávam názor, že by znázornenie plánu v agilnom prístupe bolo dokonalé. Mnohí manažéri popierajú potrebu použiť Gantt schému, keďže patrí k inému prístupu vývoja projektu. Myslím si však, že Gantt schému je možné uplatniť aj v agilnom prístupe.

Podobne ako pri šachovej partii, kde hráči podriaďujú svoju aktuálnu stratégiu hry k dosiahnutiu vopred vytýčeného cieľa, tak aj pri vývoji softvéru vieme určiť náš cieľ. Nemusím do Gantt schémy zahrnúť všetky drobné úlohy, ktoré sa môžu v projekte počas celej doby vývoja vyskytnúť. Taktiež v nej nemusím zachytiť plán na niekoľko iterácií dopredu [1]. Stačí do nej zachytiť aktuálnu iteráciu s určenými úlohami a pre ďalšie iterácie mi stačí v schéme zachytiť ich hlavné ciele. Tým nestratím čas identifikovaním všetkých čiastkových úloh v budúcnosti, no zároveň znázorním smer prác na projekte. Snáď nevyvíjame softvér bezcieľne.

Takto vytvorenú Gantt schému, ktorú dopĺňam o každú ďalšiu iteráciu viem použiť pri spätnom pohľade na projekt. Myslím si, že je prehľadnejšia ako jednoduché a nezoradené výpisy úloh, a tak je použiteľná aj v plánovaní pri agilnom prístupe.

Paralelizmus v agilnom prístupe

Agilný prístup vývoja neuvažuje o využití techniky kritickej cesty v plánovaní. Jeho cieľom je však zamestnať všetkých vývojárov pre každý šprint a efektívne využiť dostupné prostriedky. Uvažujme, že rozdelím tímu úlohy usporiadané v zozname podľa priorít zákazníka. Zo samotného zoznamu však neviem určiť závislosti úloh. Závislosti úloh viem vyjadriť len určením termínov ich začiatku a dokončenia, čo je nepraktické a nepoužiteľné. Nevidím tak kritickú cestu projektu a neviem, ktoré úlohy možno riešiť paralelne. Priority zákazníka prevyšujú priority smerovania projektu ako celku a namiesto toho aby som vývojárov zamestnal úlohami kritickej cesty, si ju len nevedomky predlžujem. Preto navrhujem explicitne zachytiť v zozname úloh ich závislosti a snažiť sa odhaliť kritickú cestu i možný paralelizmus. Takto prepojené úlohy potom už len jednoducho znázorním grafom aký využíva technika kritickej cesty.

Záver

Postupom času boli vyvinuté viaceré prístupy vývoja softvérových projektov, ktoré sa odlišujú ako v spôsobe plánovania projektu, tak aj znázornenia vytvoreného plánu. Spomedzi viacerých možností som zhodnotil tradičný prístup založený na vodopádovom modeli vývoja projektu a agilný prístup. Oba prístupy poskytujú rozličné techniky pre

znázornenie a vyhodnotenie plánu. Snažil som sa identifikovať ich výhody i nevýhody na príkladoch z vlastných skúseností. Ak sa zdá, že nami zvolený prístup vývoja projektu nám nepostačuje s technikami znázornenia plánu, navrhujem požičať si techniky z iných prístupov, prípadne ich kombinovať.

Možností ako skombinovať výhody rozličných techník môže byť viac. Základom je poznať obe stránky jednotlivých techník, ich pozitíva i negatíva, vedieť ich využiť pre svoj prospech. Učebnice manažmentu sú určite vhodným prostriedkom pre získanie vedomostí ako plánovať projekt, ale ak sa budeme držať techník pre nami zvolený prístup vývoja projektu a zatracovať ostatné, vždy budeme obeťami nedostatkov vybraného prístupu.

Nakoniec, aj tak nie je možné vytvoriť ideálny plán.

Použitá literatúra

1. Ambler, S.W.: Comparing Approaches to Budgeting and Estimating Software Development Projects. http://www.ambysoft.com/essays/comparingEstimatingApproaches.html [15-10-2012]
2. Cerpa, N., Verner, J.M.: Why Did Your Project Fail?. In: Communications of the ACM, Vol. 52, Issue 12. 2009, Dec., 12, pp. 130-134.
3. Duncan, W. R.: Schedule Development, In: A Guide to the Project Management Body of Knowledge. Project Management Institute, Upper Darby, 1996, pp. 66-71.
4. Sliger, M.: Relating PMBOK Practices to Agile Practices, Part 2 of 4. 2006, Apr., 17. http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectId=10631 [15-10-2012]
5. Christiansen, D.: The Demise of the Gantt Chart in Agile Software Projects. 2007, Jul. http://www.techdarkside.com/the-demise-of-the-gantt-chart-in-agile-software-projects [15-10-2012]

Annotation

If it does not work, it will have to

Important part of project planning is the creation of plan and its monitoring. However, activities planned for the completion of the project often take longer than was expected. There are several approaches for development of software projects and each uses different way of planning. Customer requirements for the software projects frequently change and so does the project plan even during its formation. Planning Manager must be able to respond to emerging changes in customer requirements or problems in implementation of the plan. The key thing is to visualize the plan but each approach for development offers different techniques. Since we do not know a way how to create perfect plan, there can be no perfect technique either. But we can combine the advantages of several techniques together. In the essay, we discuss combination of existing techniques for plan visualization in a traditional and agile approach.