Ako naplánovať škatuľku?

MICHAL HLAVÁČ

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

Abstrakt. Keď sa povie plánovanie softvérového projektu, mnoho ľudí si predstaví diagram s piatimi alebo šiestimi škatuľkami s názvami napríklad „Analýza a špecifikácia požiadaviek“ alebo „Integrácia a testovanie systému“. Plánovanie je však potrebné aj na oveľa nižšej úrovni granularity. Esej sa zaoberá najmä plánovaním procesov počas plynutia škatuľky implementácia v malom tíme používajúcom techniky agilného vývoja softvéru a extrémneho programovania. Pokúsim sa odpovedať na otázky: Čo je potrebné plánovať? Do akých detailov plánovať? Dôraz bude kladený na princípy plánovania pri extrémnom programovaní a na potreby malého tímu, kde sa často úloha plánovania minimalizuje a zanedbáva. Extrémne programovanie takisto klamne vzbudzuje dojem, že ide o neplánovaný a neriadený proces, kde sa pokúsim presvedčiť o opaku.

Dá sa naplánovať softvérový projekt?

Tradičnou chybou pri softvérových projektoch je skákať skôr, než ste pripravený. Keď ste pod časovým tlakom a blíži sa dátum odovzdania, je dobré „udržať projekt v pohybe“. Projekt je veľký a ťažký a je veľmi náročné zmeniť jeho smer, keď je už raz v pohybe. Preto strávte nejaký ten čas na určenie ako presne vyriešite váš problém čo najefektívnejšie a najelegantnejšie [2].

Načo vlastne vytvárať plán pre softvérový projekt? Je vôbec možné odhadnúť priebeh projektu tak aby sa tento plán dal dodržať? Prax ukázala, že vytvoriť dokonalý plán je nemožné. Nechcem teraz popierať všetky metodológie venujúce sa plánovaniu. Chcem len poukázať na to, že vytvoriť plán na začiatku projektu a odhaliť všetky neistoty a riziká nie je možné. Dá sa to však spraviť priebežne počas trvania projektu. Aj napriek tomu, že sa softvér nedá presne odhadnúť, treba robiť plány. Aj keď sa nebude dať postupovať presne podľa vytvoreného plánu, budeme mať aspoň záchytné body a približný odhad.

Projekt, ktorý nemá plán, bude nejako prebiehať a určite sa vyskytnú momenty, kde sa nebude stíhať, nebudú stačiť zdroje, alebo dokonca nebude jasné, či je potrebné niečo spraviť alebo nie. Vyskytnú sa aj situácie, kde nebude čo robiť, alebo sa bude

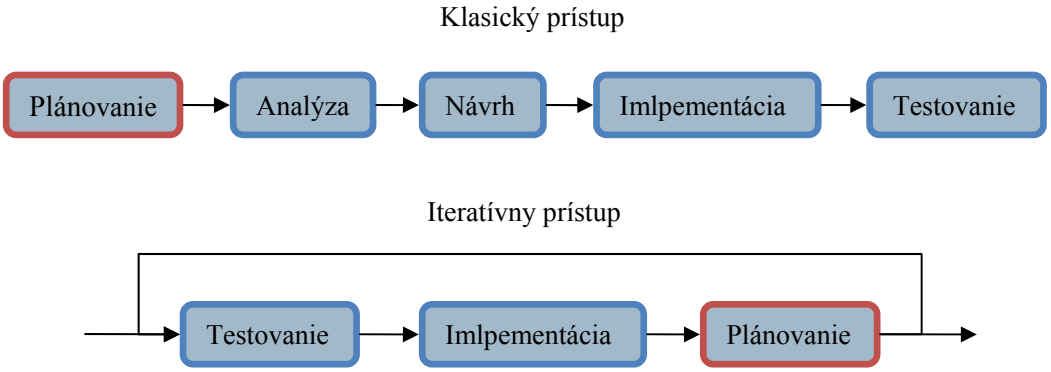
Manažment projektov softvérových a informačných systémov, október 2008, s. 1-7.

čakať na externé zdroje či poznatky. Tieto situácie ale nevnímame tak negatívne, čo je chyba, pretože sú predzvesťou ťažkého obdobia alebo dokonca zlyhania projektu.

Na druhej strane projekt s vytvoreným plánom je možné riadiť tak, aby sa čo najviac priblížil k splneniu tohto plánu. Už len to, že sa zamýšľame nad obmedzeniami a termínmi projektu, nám pomáha lepšie riadiť dostupné zdroje a identifikovať možné riziká. Plán sa priebežne modifikuje vzhľadom na nové informácie a prispôsobuje ďalším požiadavkám klienta. Priebeh projektu nemusí úplne súhlasiť s naplánovaným priebehom, napriek tomu bude tento priebeh určite priaznivejší ako v prípade, keby sme plán nemali.

eXtrémne Programovanie

Keďže sa klasický vodopádový model v praxi v podstate už nepoužíva, povedzme si radšej niečo o modernejšom prístupe vývoja softvéru ako je napríklad extrémne programovanie (XP). Extrémne programovanie je najstaršia a najpopulárnejšia metodológia agilného vývoja softvéru [5]. To, čím sa najvýraznejšie líši agilný prístup od prístupu založenom na klasickom životnom cykle softvérového systému, sú iterácie a flexibilita. Ako vidno z obrázku 1, pri agilnom iteratívnom prístupe sa stále opakujú viaceré aktivity v každej iterácii cyklu. Na druhej strane pri klasickom životnom cykle sa nedostaneme do ďalšej fázy cyklu bez toho aby sa dokončila práve aktívna fáza.



Obr. 1. Porovnanie klasického a iteratívneho prístupu vývoja softvéru.

Je pravda, že agilný prístup sa zameriava viac na aktivity ako programovanie a testovanie a aktivity určené hlavne pre analytikov a návrhárov sú viac v úzadí. Z tohto vyplýva, že tím uplatňujúci agilný prístup bude pozostávať hlavne zo všestranných programátorov a nie špecialistov na analýzu alebo návrh. Väčšinou sa ale prirodzene (niekedy aj menej prirodzene) vykryštalizuje z jedného člena líder tímu, ktorý okrem toho, že je radový programátor ako jeho tímoví kolegovia, dbá na správne dodržiavanie metodík, rieši konflikty, plánuje a manažuje v rámci svojej tímovej zóny pôsobenia, a možno len v obzore jednej iterácie.

XP zahŕňa týchto dvanásť praktík [3]:

- Malé verzie (Small Releases) – vytváranie častých verzií,
- Plánovacia hra (The Planning Game) – pomáha určiť prvky systému, ktoré majú byť hotové pre danú verziu kombinovaním obchodných priorít a technických odhadov,
- Metafora systému (System Metaphor) – vysoká úroveň abstrakcie systému, ktorá vedie vývojárov pri vývoji,
- Jednoduchý návrh (Simple Design) – nenavrhovať možné budúce vylepšenia, ale len súčasné požiadavky,
- Kontinuálne testovanie (Continuous Testing) – ku všetkému musia existovať bezchybné jednotkové testy, niekedy aj skôr ako samotný kód,
- Refaktoring (Refactoring) – úprava zdrojového kódu bez zmeny funkcionality, kvôli zlepšeniu flexibility a čitateľnosti kódu,
- Párové programovanie (Pair Programming) – programuje sa vždy vo dvojiciach,
- Kolektívne vlastníctvo kódu (Collective Code Ownership) – každý vývojár má právo zmeniť kód v ktorejkoľvek časti systému,
- Kontinuálna integrácia (Continuous integration) – integrácia zmien do systému po každej úspešnej zmene, aj viac krát za deň,
- 40 hodinový týždeň (40 hours week) – pracovať 40 hodín za týždeň,
- Prítomný zákazník (Onsite customer)– zákazník je k dispozícii počas trvania celého projektu, aby mohol zodpovedať otázky k požiadavkám systému,
- Štandardy kódovania (Coding Standards) – vysoká miera programátorskej kultúry, dodržiavanie preddefinovaných pravidiel programovania.

Nás bude zaujímať hlavne praktika „Plánovacia hra“.

Plánovacia hra

Za týmto nevinným názvom sa skrýva celá „veda“ okolo plánovania poňatá agilným prístupom extrémneho programovania [1]. Plánovacia hra je vlastne stretnutie, ktoré sa uskutočňuje raz za iteráciu, ktorá väčšinou trvá týždeň alebo dva. Na tomto stretnutí je často prítomný aj zákazník, ktorý pri XP nemusí byť nevyhnutne nejaký manažér firmy, pre ktorú robíme zákazku. Ak robíme výučbový systém pre školu, zákazníkom bude učiteľ, ktorý má aspoň približnú predstavu o tom, ako by mal systém vyzerať a čo by mal ponúkať. Zákazník je skrátka niekto, kto má užitočné informácie ohľadom požiadaviek zo strany klienta. Kľúčovými otázkami v softvérovom projekte sú: „Čo má byť spravené do najbližšieho termínu?“ a „Čo robiť teraz ako nasledovné?“. Dôraz

sa pritom kladie hlavne na riadenie projektu a nie na presný odhad, čo budeme potrebovať a ako dlho to bude trvať. Plánovanie v XP sa preto delí na tieto dva kroky:

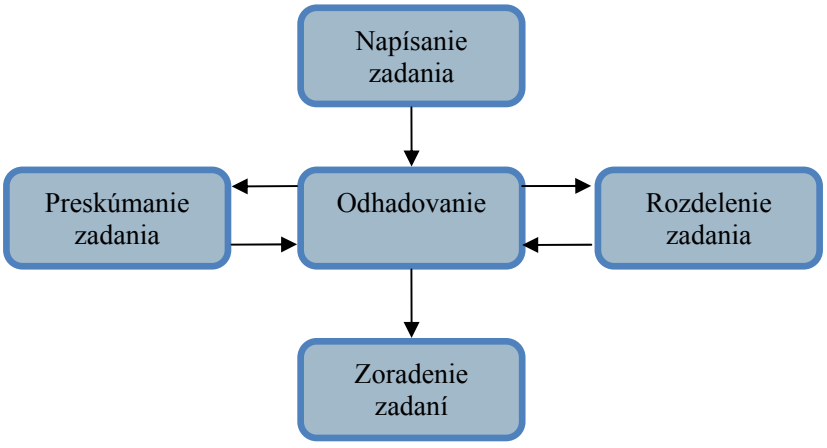
- plánovanie termínu dodania projektu
- plánovanie iterácie

Tieto plánovacie kroky sú veľmi jednoduché, ale napriek tomu poskytujú veľa informácii pre zákazníka. Navyše mu umožňujú ľahko usmerniť riadenie projektu podľa jeho predstáv. Každých pár týždňov je jasne viditeľný postup, ktorý sa udial na projekte. V XP nie je žiadnych „deväťdesiat percent hotovo“. Požiadavka sa buď spravila, alebo nie. Toto zameranie na viditeľnosť vytvára v XP zvláštny paradox. Na jednej strane je postup taký viditeľný, že zákazník môže kedykoľvek prerušiť projekt, ak mu nevyhovuje jeho výstup. Na druhej strane postup je taký viditeľný a schopnosť rozhodnúť, čo sa má spraviť ako nasledujúce je taká silná, že XP projekty smerujú k dodaniu toho, čo je naozaj potrebné, pri menšom tlaku a strese.

Plánovanie termínu dodania projektu

Pri plánovaní termínu ďalšej verzie prezentuje zákazník svoje zadania (stories) na funkcionalitu systému a vývojári sa pokúšajú odhadnúť ich zložitosť. Na obrázku 2 vidíme hlavné činnosti, ktoré sú súčasťou tejto fázy plánovania.

Zadania sa spravidla píšu na malé kartičky (story cards), aby bolo zaručené, že sa tam fyzicky nezmestia veľké zadania. Takáto kartička obsahuje názov a malý odsek, kde je opísaná požiadavka vo forme zadania. Dobré zadanie je, ak sa dá na neho odhadnúť jeden až tri body (story point). „Bod“ predstavuje jednotku výkonu tímu. Vo všeobecnosti jeden bod znamená, že na danom zadaní bude tím pracovať jeden týždeň. Záleží však na tíme, koľko bodov dokážu spraviť za jeden týždeň alebo za jednu iteráciu. Ak je to potrebné, na kartičke môžu byť pripojené aj podporné materiály. Treba dbať však na to, že zadanie nemusí obsahovať všetky detaily želanej požiadavky. Je to len podporný nástroj pri diskusii.



Obr. 2. Hlavné činnosti vo fáze plánovania termínu dodania projektu.

Ak je zadanie príliš veľké, vývojári ho vrátia zákazníkovi, aby ho rozdelil. Dôvodom pre rozdelenie môže byť zvýšenie flexibility implementácie alebo je zadanie jednoducho príliš veľké na to, aby sa preň dal spraviť rozumný odhad. Aj pre zákazníka sú menšie zadania lepšie, pretože tým znižuje rozsah možného zle implementovaného zadania. Ak zadanie nie sú schopní vývojári odhadnúť a nespočíva to vo veľkosti zadania, musí prebehnúť fáza preskúmania, kde sa analyzujú možné riešenia zadania.

Zákazník potom zoradí kartičky podľa dôležitosti, kde najdôležitejšie zadanie je navrchu. Ak máme k dispozícii odhady zložitosti požiadaviek a ich prioritné poradie, môže zákazník následne vytýčiť plán projektu. Musí teda rozhodnúť, ktoré zadania sa zahrnú do projektu a kedy bude termín jeho ukončenia.

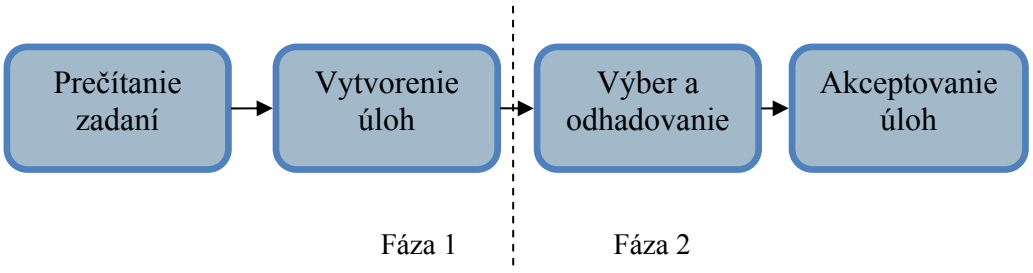
Spočiatku sú termíny veľmi nepresné a inak to ani nemôže byť. Pokiaľ tím ešte nezačal pracovať na samotnom projekte, nie sú tieto odhady ani priority pevne podložené. Aby bolo možné ďalej pracovať s týmto plánom, je nutné, aby sa k nemu tím pravidelne vracal a upravoval ho. Plánovanie termínu dodania väčšinou prebieha v prvých štyroch týždňoch projektu.

Vďaka malým zadaniam a bodom má zákazník dobré nástroje na flexibilné ovplyvňovanie riadenia projektu. Ak sa mu nepáči čo vidí, môže zmeniť zadania, ustanoviť nový dátum ukončenia, modifikovať tím, alebo ho úplne vymeniť za iný.

Plánovanie iterácie

Ako sme si povedali, jedna iterácia trvá väčšinou týždeň alebo dva. Na konci každej iterácie sa musí odovzdať nejaká funkčná a užitočná časť projektu. Počas prvého dňa iterácie sa tím rozhodne, na ktoré zadania sa zameria. Tím už dosiahol nejaký počet bodov v predošlej iterácii. Pri prvej iterácii sa zvyčajne myslí na to, že sa projekt len rozbieha a počíta sa s jednou tretinou bodu za týždeň.

Zákazník si vyberie zadania, ktoré mal pripravené z predošlej časti plánovania a určí, ktoré by chcel aby boli implementované v danej iterácii. Zadanie nemusia byť v rovnakom poradí ako boli prezentované v pláne dodania projektu. Zákazník môže požadovať implementáciu v ľubovoľnom poradí. Pre vývojárov by to nemal byť problém, pretože jednotlivé zadania predstavujú často implementačne oddeliteľné časti. Zákazník môže navyše pridať nové zadania, ak je ochotný dať tímu čas na ich odhadnutie. Na obrázku 3 vidíme, že plánovanie iterácie sa delí ešte ďalej na dve fázy.



Obr. 3. Plánovanie iterácie.

V prvej fáze hry sa prečítajú všetky kartičky zo zadaniami a vývojári spoločne so zákazníkom určujú úlohy, ktoré by sa mohli týkať daných zadaní. Na konci tejto fázy už majú vývojári k dispozícii veľa úloh zodpovedajúcich zadaniam, ktoré boli vybrané. Jedna úloha nemusí zodpovedať práve jednému zadaniu, ale môže podporiť aj viaceré zadania.

V druhej fáze sa až tak výrazne neprejavuje funkcia zákazníka. Vývojári si rozdelia úlohy a odhadnú ich cenu. Teraz to však robia na oveľa nižšej úrovni granularity ako v plánovaní termínu dodania projektu. Vyberú a priradia si úlohy medzi sebou. Aj v tejto fáze môžu nastať nezrovnalosti ak sa nepodarí na nikoho „napasovať“ nejakú úlohu. V tom prípade musí zákazník zadanie, ktorého sa daná úloha týkala, odložiť alebo rozdeliť. Môže sa takisto stať, že má tím viac času ako úloh a vtedy je nutné pridať nejaké zadania do konkrétnej iterácie.

Prihliadnuc na množstvo úsilia, ktoré bolo vynaložené na ukončenie predchádzajúcej iterácie, sa vývojári zaviažu k naplneniu vybraných požiadaviek do ďalšej iterácie.

Nakoniec vytvoril tím spoločne so zákazníkom plán iterácie, ktorý pokrýva celú množinu zadaní požadovanej zákazníkom.

Záver

Plánovanie je neoddeliteľnou a dôležitou súčasťou každého softvérového (a nie len softvérového) projektu. V súčasnosti už nevystupuje ako jedna z prvotných fáz životného cyklu projektu, ktorá má koniec ešte pred spustením samotného projektu. Je to skôr priebežný proces, počas trvania skoro celého projektu, kde sa v iteráciách postupne zjemňuje a upresňuje.

Zodpovednosť za plán nemajú len projektoví manažéri, ale aj ľudia postavení nižšie v hierarchii projektu. Bez nich by sa plán nielenže nedal naplniť, ale ani vytvoriť v dostatočnej kvalite na to, aby sa dal používať. Ak sa pri plánovaní zabudneme spýtať špecialistov na ich názor, napríklad vedúceho testovacieho oddelenia, môže byť plán pekný, jasný a prehľadný, ale určite nie blízky realite a preto nepoužiteľný.

Moderný a už aj v značnej miere používaný model vedenia softvérového projektu nám ponúka extrémne programovanie. Plánovacia hra, ktorá je jedným z hlavných princípov extrémneho programovania, je skvelým príkladom ako plánovať v malých tímoch pomerne rýchlo sa vyvíjajúce sa projekty.

Princípy agilného vývoja softvéru sa ale ešte stále neuchytili všade a niekedy býva prechod na XP spojený s nedôverou a pochybnosťami. Tieto princípy sa možno zdajú zvláštne a príliš jednoduché na to aby fungovali, ale napriek tomu by som rád odporučil tento pohľad na vedenie projektu každému tímu. Existuje hra [4], ktorou sa dá simulovať XP projekt bez toho, aby bolo nutné „obetovať“ reálny projekt. Uplatnením XP sa môže zlepšiť kvalita ľubovoľného tímu a zaradiť sa tak medzi úspešné tímy, ktoré dodávajú kvalitný softvér svojim klientom.

Použitá literatúra

1. Jeffries, R.: What is Extreme Programming? (11.8.2001) http://www.xprogramming.com/xpmag/whatisxp.htm (prístup 15.10.2008)
2. Jenkins, N.: *A Project Management Primer or "a guide to making projects work (v2.0)"*, 2006.
3. Kuppuswami, S., Vivekanandan, K., Ramaswamy, P., Rodrigues, P.: The Effects of Individual XP Practices on Software Development Effort, *ACM SIGSOFT Software Engineering Notes*, Vol.28, Issue 6, 2003.
4. Marchesi, M., Succi, G., Wells, D., Williams, L.: *Extreme Programming Perspectives*, Addison Wesley, Chapter 27, 2002.
5. Noble, J., Marshall, S., Marshall, S., Biddle, R.: Less Extreme Programming, In: *Proc. of the sixth conference on Australasian computing education*,Vol. 30, Australian Computer Society, Inc., Dunedin, New Zealand, 2004, 217-226.

Annotation

How to plan a box?

When talking about planning of a software project, many people think about a diagram with five or six boxes with captions on them like "Analysis and requirement specification" or "Integration and testing". Planning is also necessary in a much lower level of granularity. This essay particularly proceeds of planning of processes in the duration of the box implementation in a small team that uses techniques of agile software development and extreme programming. I will try to answer these questions: What has to be planned? How deep should planning go? The emphasis will be put on the principals of scheduling process in extreme programming and also on the needs of a small team where the task of scheduling is commonly marginalized. Extreme programming also invokes a misleading impression that it is an uncontrolled process, where I try to convince you of the opposite.