Plánovanie agilného vývoja softvéru

JURAJ KOLLÁR

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jkollar33[zavináč]gmail[.]com

Abstrakt. Častou príčinou zlyhania mnohých projektov je zlé, prípadne žiadne plánovanie. Preto sa asi všetci zhodneme na tom, že dobré plánovanie je základom úspechu každého projektu. V súčasnej dobe sú vyvíjané systémy čoraz komplexnejšie a požiadavky sa menia veľmi často. Z tohto dôvodu sa klasický vodopádový model ukázal ako nevhodný. Riešení je viacero, no v posledných rokoch ide najmä o rôzne agilné metódy vývoja ako XP, Scrum alebo Crystal. Tieto metódy majú spoločnú črtu, a to iteratívny spôsob vývoja a časté uvoľňovanie verzii, vďaka čomu sa dá prispôsobovať častým zmenám a nepresnostiam v požiadavkách. Otázkou však zostáva, ako sa dá takýto spôsob vývoja plánovať. Táto esej sa snaží poskytnúť odpoveď na túto otázku ako aj ďalšie odporúčania a názory ako vytvárať plán pre agilný vývoj softvéru.

Časy sa menia a ani vodopád už nie je to, čo býval

Vývoj softvéru sprevádzajú od jeho začiatku neúspechy. Väčšina z týchto neúspechov pramení zo zlého plánovania, resp. neschopnosti reagovať na zmeny [6]. Preto v dnešnej dobe už málokto podceňuje túto činnosť, ktorá do značnej miery ovplyvňuje úspech alebo neúspech projektu, a prikladá sa na ňu veľký dôraz. Prístupov k plánovaniu je však viacero.

V minulosti bol zaužívaným modelom tzv. vodopádový model. Ten predpokladal, že každá fáza vývoja softvéru (analýza, návrh, implementácia atď.) musí nasledovať až po skončení tej predchádzajúcej. Plánovanie takéhoto vývoja potom vyzeralo tak, že na začiatku projektu sa spravil čo najdetailnejší plán a tím vývojárov sa ho snažil striktne dodržiavať.

Časy sa však menia a tento model sa ukázal ako ťažkopádny. Jednou z príčin jeho neúspechu bol predpoklad, že dobrý plán sa dá spraviť hneď na začiatku a ďalej sa už nemá meniť. Z praxe však vieme, že požiadavky sa zvyknú meniť pomerne často a aj ten najdetailnejší odhad sa môže stať nereálnym.

Jedným z riešení sa stal špirálový model alebo celý proces ako napr. RUP (Rational Unified Process). V posledných rokoch vznikajú rôzne iné, tzv. agilné, metódy vývoja. Či už ide o XP (eXtreme Programming) alebo Scrum, jednu črtu majú

Manažment projektov softvérových a informačných systémov, október 2008, s. 1-8.

spoločnú, a tou je iteratívny a inkrementálny spôsob vývoja s častým uvoľňovaním verzií (angl. release). Zjednodušene ide o to, že softvér sa vyvíja v krátkych iteráciách, zameraných na jednu konkrétnu činnosti, prípadne požiadavku [7]. Pri takomto spôsobe vývoja však treba zmeniť aj tradičný pohľad na plánovanie.

Plánovanie iteratívneho projektu v RUP

Na začiatok, keď idem hovoriť o RUP, treba poznamenať, že správne by sa malo hovoriť o UP (angl. unified process), pretože RUP je konkrétny produkt spoločnosti IBM. Pojem RUP je však natoľko zaužívaný, že ho v tejto eseji budem používať ako synonymum UP.

Projekty, ktoré sa vyvíjajú podľa RUP, majú zväčša 2 typy plánov: tzv. hrubozrnný (angl. coarse-grained) a jemnozrnný (angl. fine-grained) [4]. Hrubý plán odhaduje celkovú dĺžku projektu s významnými míľnikmi, no nejde viac do hĺbky. Zameriava sa skôr na fázy projektu a ciele jednotlivých iterácií. Jemné plány potom predstavujú plány jednotlivých iterácií.

Hrubý plán v RUP predstavujú jednotlivé fázy počas životného cyklu projektu. Medzi ne patrí počiatok, rozpracovanie, konštrukcia a premena (angl. inception, elaboration, construction, transition). Skončenie každej z týchto fáz (aj keď o konci sa nedá presne hovoriť, keďže fázy sa prekrývajú) predstavuje dôležitý míľnik vo vývoji projektu. Termíny týchto míľnikov by sa mali dodržať.

Vďaka tomu, že v danej iterácii presne vieme, čo ideme robiť, môže byť aj jej plán podrobnejší. S výhodou môžeme potom použiť klasické plánovacie metódy ako napríklad Ganttove diagramy. Taktiež sa jednoduchšie rozdeľujú úlohy medzi vývojárov a možno definovať presné termíny, kedy má byť čo hotové.

Ďalšími parametrami, ktoré treba naplánovať je dĺžka a počet iterácií. Tieto dva parametre závisia od počtu ľudí v tíme ako aj od povahy projektu. Podľa počtu ľudí v tíme by sa mohla dĺžka iterácií stanoviť nasledovne [4]:

- 5 ľudí: jeden týždeň
- 20 ľudí: 3-4 týždne
- 40 ľudí: 8 týždňov

Počet iterácií pre stredne veľký projekt môžeme pre jednotlivé fázy stanoviť nasledovne:

- počiatok: jedna iterácia
- rozpracovanie: dve iterácie
- konštrukcia: dve iterácie
- premena: jedna iterácia

Dobrým zvykom je dať jednotlivým iteráciám názov, čo umožní lepšie porozumenie, čo sa v danej iterácii vyvíja.

Scrum ako agilná metóda plánovania

Keď som prvýkrát počul o metóde Scrum, nazdával som sa, že ide o metódu vývoja, no neskôr som prišiel na to, že je to skôr prístup k plánovaniu, ktorý nehovorí o tom, akým štýlom sa má softvér programovať. Teda nemusí napríklad platiť, že ak používame Scrum, tak musíme taktiež vyvíjať technikami XP. Treba tiež rozlišovať dva, na prvý pohľad rovnaké pojmy: plánovanie agilného vývoja a agilné plánovanie vývoja. Kým pri plánovaní agilného vývoja je agilný vývoj, tak pri agilnom plánovaní vývoja je agilný práve proces plánovania a nie samotný proces vývoja.

Scrum by som zaradil medzi agilné metódy plánovania. Prečo? Lebo sleduje rôzne agilné princípy definované agilným manifestom ako [1, 5]:

- *jednotlivci a ich interakcie* sú dôležitejšie ako procesy a nástroje
- *fungujúci softvér* je viac ako vyčerpávajúca dokumentácia
- *spolupráca so zákazníkom* je lepšia ako vyjednávanie na zmluve
- *prispôsobovanie zmenám* je lepšie ako striktné držanie sa plánu

Takže čo je to teda ten Scrum? Je to agilná metóda plánovania, ktorá definuje princípy, ktorými sa má vývoj riadiť, vymedzuje roly v projekte, používa vlastnú terminológiu atď. Dbá sa hlavne na tímovú prácu, aktívnu účasť zákazníka a schopnosť reagovať na zmeny. V nasledujúcich odsekoch opíšem jednotlivé činnosti, roly a pojmy v Scrume [7].

Tzv. zoznam nedorobenej práce na produkte (angl. Product Backlog) je priorizovaný zoznam požiadaviek projektu, ale nie len nich, ale aj ďalších vecí súvisiacich s projektom. O tento zoznam sa stará tzv. vlastník produktu (angl. Product Owner). Jeho úlohou je určovať priority jednotlivých požiadaviek, ktoré sa samozrejme môžu a budú počas projektu meniť. Mal by to byť človek, ktorý vie, čo je pre daný produkt najdôležitejšie. Ideálne by podľa môjho názoru mal byť týmto človekom sám zákazník, resp. zákazníkom poverená osoba.

Ďalšími rolami sú tzv. tím vývojárov (angl. Scrum Team) a „vodca tímu“ (angl. Scrum Master). Vodca tímu by sa dal prirovnať ku projektovému manažérovi - jeho úlohou je podporovať tím, viesť ho a odstraňovať vzniknuté prekážky. Ďalej sa pozrieme na samotný proces vývoja v Scrume.

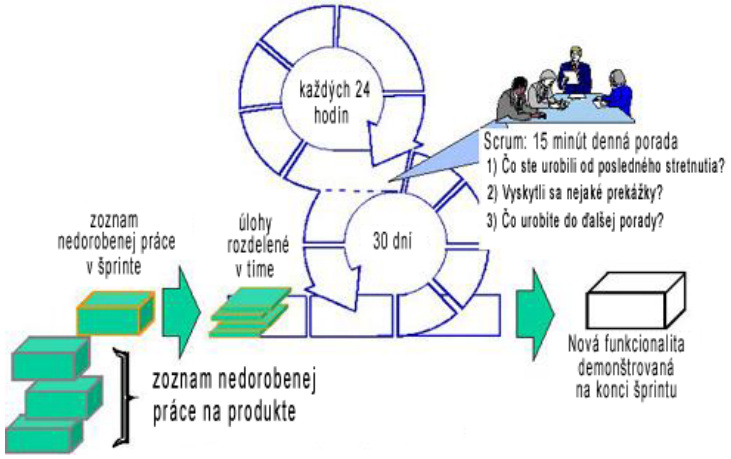
Prvou úlohou je zostaviť zoznam nedorobenej práce. Touto činnosťou sa primárne zaoberá vlastník produktu, ale keďže ide o agilný prístup, každý z členov tímu môže do tohto zoznamu pridať hocikedy hocičo. Vlastník produktu potom pridelí jednotlivým položkám priority a podľa nich tento zoznam zotriedi.

Ďalšou úlohou je spraviť odhad. Tu sa dá použiť bežný prístup zhora nadol. Používajú sa však aj viac „vyšinuté“ prístupy, napríklad neodhadovať v časových jednotkách, ale v relatívnych číslach ako napr. Fibonacciho postupnosť [8]. Pomáha to odhadnúť relatívnu náročnosť jednej požiadavky oproti inej. Ďalšou z agilných metód pri plánovaní je zapojiť doň celý tím tzv. metódou Delphi, ktorá sa tu nazýva „plánovací poker“ (angl. planning poker). Ide o to, že každý z členov tímu napíše svoj odhad na vlastný papier, aby nebol ovplyvnený kolegami a potom sa papiere naraz ukážu a diskutujú sa jednotlivé odhady.

Jednotlivé iterácie sa v Scrume nazývajú šprinty (angl. Sprint). Šprinty by mali byť čo najkratšie, odporúčaná dĺžka je týždeň až jeden mesiac. Dĺžka šprintu by mala byť konštantná, aby si tím zvykol na istý rytmus. Pred každým šprintom treba spraviť stretnutie (angl. Sprint Meeting), na ktorom sa zúčastnia všetci členovia tímu. Vyberú sa položky, ktoré sa budú vykonávať v danom šprinte, čím sa vytvorí tzv. zoznam nedorobenej práce v šprinte (angl. Sprint Backlog). Typicky ide o položky s najväčšou prioritou v zozname nedorobenej práce na produkte. Vlastník produktu vysvetlí jednotlivé požiadavky napr. formou tzv. používateľských príbehov (angl. user stories) prevzatých z XP. Sú to požiadavky v tvare: „Ako [typ používateľa], chcem [niečo, funkcionalitu], aby som dosiahol [nejaký cieľ]“.

Teraz zostáva už len naplánovať samotný šprint. Požiadavky sa rozčlenia na úlohy, môže ísť pokojne aj o tie klasické ako analýza, návrh, implementácia, testovanie. Odhadne sa dĺžka ich trvania, opäť sa na tomto odhade podieľa celý tím. Nakoniec si jednotliví členovia tieto úlohy rozdelia.

Začne sa „šprintovať“, teda vyvíjať za účelom dosiahnutia cieľa. Dôležitým faktom je, že v takomto krátkom časovom úseku sa musí dodržať termín daný dĺžkou šprintu, ktorý je *fixný*. Ak sa predsa len nejaká úloha nestíha splniť, jednoducho sa odstráni zo zoznamu daného šprintu. Ďalším dôležitým faktom je, že požiadavky sú počas šprintu „zmrazené“, teda každý pracuje na dohodnutých úlohách a nemali by pribúdať nové. Na konci šprintu sa opäť spraví stretnutie, kde sa predstaví nová funkcionalita, ktorá vznikla. Vizuálne si to môžeme predstaviť ako na **Obr. 1**.



Obr. 1. Scrum vo vizuálnej podobe [9].

Jednou z vecí, ktoré sa mi na Scrume páčia, sú krátke, tzv. denné stretnutia (angl. Daily Scrum), kde musia byť opäť prítomní všetci členovia tímu, hlavne vlastník produktu. Všetci stoja v kruhu okolo tabule a každý hovorí, čo sa mu podarilo od posledného stretnutia, čo plánuje urobiť do ďalšieho, aké má prekážky pri plnení úloh, prípadne diskutuje svoje úlohy s vlastníkom produktu. Takto sa postupne

vystriedajú všetci, naraz však hovorí vždy len jeden. Pripomína to „bojové porady“ z amerického futbalu, kde hráči stoja v kruhu a preberajú taktiku, odtiaľ aj pochádza pojem „scrum“, ktorý označuje túto „bojovú poradu“. Toto stretnutie sa deje každý deň, trvá zhruba 15 minút. Zvoláva ho a moderuje vodca tímu.

Tieto cykly - šprinty sa neustále opakujú a postupne tak odstraňujú položky zo zoznamu nedorobenej práce na produkte podľa potreby a preferencií zákazníka, kým tento nie je s produktom spokojný.

Ako si vybrať?

Na začiatok treba povedať, že ani jedna z uvedených metód nie je tzv. strieborný náboj a má svoje pre aj proti [2].

RUP je veľmi dobre definovaný ucelený proces, ktorý presne definuje celý proces vývoja softvéru. Je to akási „šablóna“, definuje presne postup činností, roly, dokonca existuje pre každú činnosť šablóna dokumentu, kde sa má daná činnosť opísať (analýza, návrh, atď.). Oproti agilným metódam ako Scrum alebo XP sa kladie veľký dôraz na analýzu a návrh, ktoré sa dejú v počiatočných fázach projektu.

Čo sa týka samotného plánovania, v RUP sa robí plán hneď na začiatku, hoci ide len o hrubý plán. V ňom sú dôležité tzv. míľniky (angl. milestone). Po prvom míľniku, ktorý vznikne po analýze, sa ešte môžeme rozhodnúť projekt ukončiť. Je to niečo ako štúdia vhodnosti. Ďalším míľnikom je ukončenie návrhu, tu sa tiež ešte môžeme rozhodnúť v projekte nepokračovať, alebo spraviť nový návrh. Hoci sa v RUP podobne ako v agilných metódach využíva iteratívny spôsob vývoja, prvé výsledky vidí zákazník až pomerne neskoro [4].

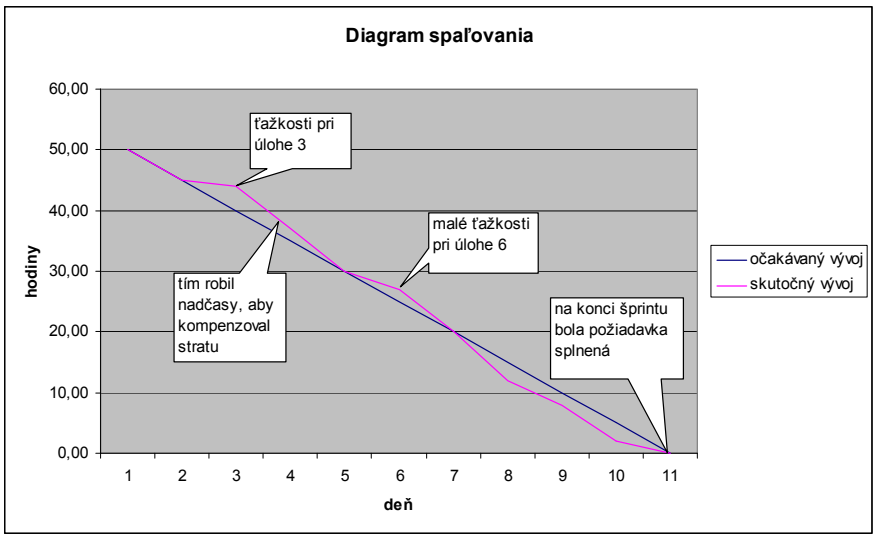
Agilné metódy vyznávajú skôr iné princípy. Na začiatku sa príliš neplánuje a nekladie sa ani veľký dôraz na analýzu a návrh. Z môjho pohľadu je toto jedna zo slabín agilných metód, kde sa postupuje požiadavka po požiadavke, čo bez dôkladného návrhu môže viesť k značnej nekvalite kódu, pretože sa stále nabaľuje nová funkcionalita a pôvodný, slabo navrhnutý kód sa stáva príliš zložitým. Aj na tento neduh však existujú protizbrane, napríklad refaktoring v metóde XP.

Využívanie agilného prístupu má však veľa iných kvalít, ktoré, si myslím, stoja za zmienku. Jednou z nich je iteratívny vývoj (šprinty) a časté pribúdanie novej funkcionality. Aj keď to na prvý pohľad nemusí byť jasné, tak tento prístup umožňuje aj skoré testovanie - to je totiž súčasťou každého šprintu [7].

Najväčší prínos agilných metód však vidím inde. Jednou z kľúčových vlastností pre projektového manažéra, ale aj zákazníka je viditeľnosť. Vďaka agilným metódam ako Scrum, hneď vidíme, čo už je hotové a čo ešte nie (zoznam nedorobenej práce na produkte). K lepšej viditeľnosti tiež prispievajú denné krátke porady. Eliminuje sa tým známy syndróm 90% hotovo [3]. Ďalšou z veľmi dôležitých vlastností agilných metód je prispôsobivosť a otvorenosť pre zmeny. Kým v procese ako je RUP sa kladie dôraz na dodržiavanie vopred definovaného plánu, pri agilných metódach vidíme po každom cykle čo máme hotové a čo ešte nie [8]. Tu môže zasiahnuť zákazník a zmeniť priority v požiadavkách alebo pridať aj nové, prípadne niektoré zrušiť alebo odložiť. Aj aktívna účasť zákazníka pri vývoji prispieva k viditeľnosti a väčšej flexibilite.

Rozdiel medzi agilnými metódami a tradičnými je aj v rôznom prístupe k tvorbe rozpočtu, resp. dodržaniu termínu dodania softvéru. Pri RUP sa kladie dôraz na implementáciu všetkých požiadaviek, pričom termíny sa môžu posunúť a rozpočet zvýšiť, zatiaľ čo pri agilnom vývoji sa kladie dôraz na dodržanie termínov a rozpočtu, pričom požiadavky sa implementujú podľa potreby, teda niektoré menej podstatné sa môžu vynechať za účelom vtesnania sa do rozpočtu. Nakoniec zákazník dostane presne to, čo chce, pretože sa aktívne podieľa na vývoji, určuje čo sa bude a čo sa nebude implementovať a hneď vidí aj výsledky vďaka krátkym iteráciám a môže sa vyjadriť, či je spokojný alebo to treba prerobiť.

Jednou z dôležitých činností, ktoré sa týkajú plánovania, je aj sledovanie, ako sa plán plní. Na toto má napríklad Scrum skvelý nástroj, ktorý sa nazýva „diagram spaľovania“ (angl. Burndown Chart) [7]. Je to diagram závislosti potrebnej práce v hodinách od času v dňoch. Sú v ňom zakreslené dva grafy - jedným je očakávaný vývoj a druhým je skutočnosť, ako sa plní (spaľuje) daná úloha. Každý vývojár má vlastný diagram a na dennej báze doňho zaznačuje postup práce, prípadne aj doplňuje komentáre k vývoju grafu. Príklad takéhoto grafu sa nachádza na **Obr. 2**



Obr. 2. Diagram spaľovania.

Agilné plánovanie v tímovom projekte

Tímový projekt je predmet, ktorého účelom je vytvoriť projekt menšieho rozsahu (v porovnaní s komerčnými projektmi) a vyskúšať si tímovú prácu. Každý tím sa skladá z 5 alebo 6 študentov. Použitie agilných metód by preto na prvý pohľad mohol byť celkom zaujímavý nápad. Napriek tomu, že tieto metódy pokladám za dobré

a prínosné, v tomto prípade by som ich však neodporúčal. Dôvod je hlavne ten, že študenti majú okrem tímového projektu aj iné povinnosti, a preto nemôžu tráviť čas každý deň len ním. Taktiež môžu mať iné rozvrhy, takže každodenné stretávanie je prakticky nemožné. Rovnako sa ťažko určuje, kto je zákazník, ktorý by priorizoval požiadavky.

Odporúčal by som ale vziať si z agilných metód aspoň niečo, napríklad vyvíjať v krátkych iteráciách - napríklad o dĺžke jeden týždeň, plánovať každý týždeň a často konzultovať s vedúcim pedagógom.

Záver

Agilné metódy plánovania prinášajú mnohé zaujímavé pohľady na plánovanie vývoja softvéru, cenia si hlavne tímovú prácu, komunikáciu, vítajú zmeny a oceňujú aktívnu účasť zákazníka, čo v konečnom dôsledku prispieva k jeho spokojnosti. Tieto metódy sú dobrá voľba pre malé tímy - zhruba 5 - 10 ľudí, ktorí pracujú v tesnej blízkosti, najlepšie v rovnakej kancelárii. Pre väčšie tímy sa však dajú použiť len s ťažkosťami a lepšie je v takomto prípade siahnuť po tradičnejších metódach, nie však po vodopádovom modeli. Ak sa teda rozhodneme použiť metódu Scrum alebo inú agilnú metódu, odporúča sa mať v tíme človeka, ktorý už má s ňou skúsenosti, pretože v opačnom prípade by jej využitie mohlo mať nepriaznivý efekt.

Na záver si dovolím krátky citát na zamyslenie, ktorého autorom je vedec Charles Darwin: „*Prežije nie ten druh, ktorý je najsilnejší, ale ten, ktorý sa dokáže najviac prispôsobiť*“.

Použitá literatúra

1. Abdel-Hamid, T. 1988. Understanding the "90% Syndrome" in: Software Project Management: A Simulation-Based Case Study. Journal of Systems and Software. s. 319-330.
2. Beck, K. et al. 2001. Manifesto for agile software development. [citované 17.10.2008] Dostupné z <http://agilemanifesto.org/>
3. Brooks, F. P.: No Silver Bullet - Essence and Accident in Software Engineering. In: Proceedings of the IFIP Tenth World Computing Conference, 1986, s. 1069-1076
4. Kruchten, P. 2002. Planning an Iterative Project. [citované 17.10.2008] Dostupné z <http://www.ibm.com/developerworks/rational/library/2831.html>
5. Maurer F. – Melnik G.: Agile methods: moving towards the mainstream of the software industry. In: ICSE '06: Proceedings of the 28th international conference on Software engineering: ACM, May 2006, s. 1057-1058
6. Waters K. Why most IT Projects Fail. And How Agile Principles Help. [citované 17.10.2008] Dostupné z <http://www.agile-software-development.com/2007/08/why-most-it-projects-fail-and-how-agile.html>

7. Waters K. How To Implement Scrum In 10 Easy Steps. [citované 17.10.2008] Dostupné z <http://www.agile-software-development.com/2007/09/how-to-implement-scrum-in-10-easy-steps.html>
8. Waters K. Agile Project Planning. [citované 17.10.2008] Dostupné z <http://www.agile-software-development.com/2008/08/agile-project-planning.html>
9. Agile Techno Solutions: The Scrum Process Diagram. [citované 17.10.2008] Dostupné z <http://agile.vn/ItemDetails.aspx?ItemID=17&CategoryID=5>

Annotation

Planning of agile software development

The reason why many projects fail is bad or no planning. Therefore we may all agree that good planning is fundamental for every project. Software systems developed in present are very complicated and requirements are changing very often. This is the reason why traditional waterfall model is not suitable any more. There are few solutions but recently various agile methods such as XP, Scrum or Crystal emerged. They have common features such as iterative development and often releasing which allows us to adapt to changes and uncertainties in requirements. Questionable is how can we plan such way of development. This essay is trying to answer to this question and give my recommendations and opinions on how to plan agile software development.