Vplyv prístupu k vývoju softvéru na tvorbu plánu

BC. ALEXANDER ŠIMKO

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
simko04[zavináč]student.fiit.stuba[.]sk

Abstrakt. Plánovanie má svoje miesto pri vytváraní ľubovoľného softvérového produktu. Prináša výhody spojené s lepším odhadom rozsahu projektu, jeho ceny, dĺžky trvania, prideľovaním úloh a podobne. V súčasnosti nikto nepopiera potrebu plánovania. Avšak každému človeku, ktorý stojí pred úlohou zostrojiť plán, napadnú otázky: „Aký podrobný má plán byť? Aké obdobie má pokrývať?“ Tak ako použitý prístup ovplyvňuje rôzne aspekty vývoja softvéru, ovplyvňuje rovnako aj špecifiká projektového plánu. Táto esej sa venuje práve vplyvom použitého prístupu na plánovanie a rozoberá uvedené otázky.

Úvod

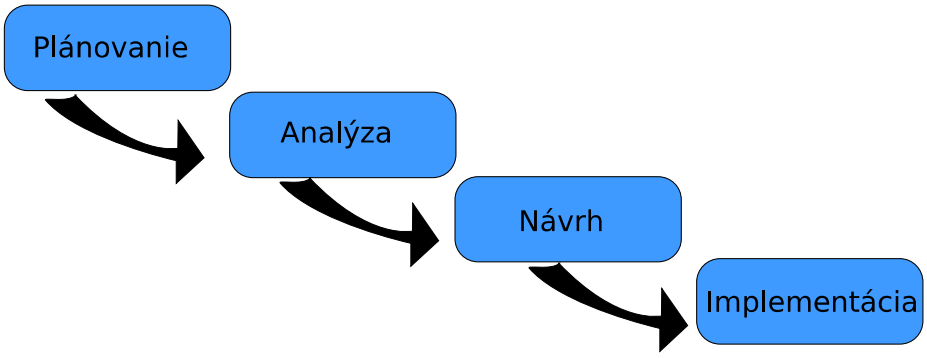
Podmienky a prostredie, v ktorom sa softvér vytvára, sa rôznia. Nech sú však okolnosti tvorby softvéru akokoľvek odlišné, spoločným menovateľom stále ostáva neznámo. Môžu ním byť rôzne veci: ochorenie člena vývojového tímu, odhalenie chyby v používanom nástroji, či vrtochy zákazníkov. Všetky tieto veci vplývajú na dodržanie stanoveného plánu, a tým pádom na včasnosť dokončenia softvérového produktu, jeho kvalitu a cenu. Metodiky a prístupy k vývoju softvéru sa snažia zvoliť práve taký spôsob práce, ktorý považujú za najvhodnejší na dosiahnutie stanoveného cieľa. Použitý prístup má potom priamy vplyv na tvorbu plánu.

Prístupy k vývoju softvéru

Plánom riadený vývoj softvéru

Rôzne plánom riadené prístupy predpokladajú, že tvorba softvéru môže byť úplne predpovedaná a naplánovaná. Často používajú vodopádový model životného cyklu softvéru (**Obr. 1**.).

Manažment v softvérovom inžinierstve, október 2007, s. 1-1.



Obr. 1. Vodopádový model vývoja softvéru [3]

Tvorba softvéru je rozdelená do jednotlivých presne definovaných fáz, ktoré nasledujú za sebou. Akonáhle je jedna fáza ukončená, už sa k nej nevraciame. Predpokladáme, že sa nič mimo našich úvah a plánov nevyskytne. V prípade, že sa pri tvorbe softvéru odkloníme od stanovenej cesty, celý plán prestáva byť platný.

Agilný prístup

Agilný prístup - tak ako je uvedený v Manifeste agilného vývoja [2] - kladie dôraz na:

- interakciu vývojového tímu so zákazníkom
- krátke iteračné cykly
- prispôsobovanie sa zmenám

a to aj v neskorších fázach vývoja softvéru.

Tento prístup už vo svojich základných princípoch počíta so zmenou ako s niečím nevyhnutným a prirodzeným. Ako nástroj na vyrovnanie sa s touto skutočnosťou zavádza krátke cykly. Ich zavedením určuje, aby neboli úlohy plánované podrobne na dlhý časový úsek, ale aby sa pozornosť zameriavala na obdobie najbližšieho cyklu.

Skúsenosti z reálnej praxe

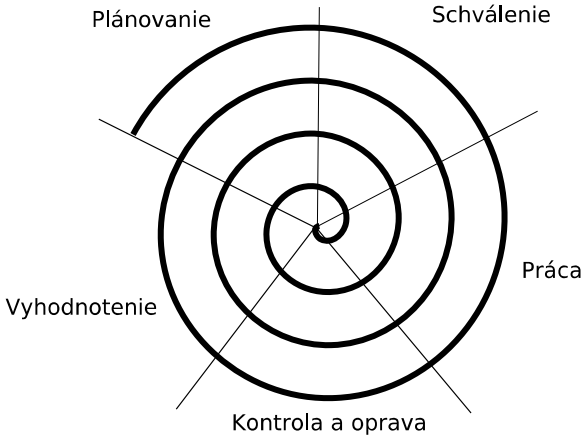
Tvorba softvéru v malom tíme

Rettig a Simos vo svojom článku [5] popisujú skúsenosti, ktoré získali zavedením nového procesu pre plánovanie a riadenie vývoja softvéru. Podľa autorov je komplexný systém ťažké správne špecifikovať na prvýkrát, častokrát sa to nepodarí ani druhýkrát. Práve preto sa rozhodli použiť iteratívnu stratégiu vývoja softvéru.

Vo svojom prístupe rozdelili úlohu na viacero fáz, každá z nich riadená špeciálnym životným cyklom:

- zlátanina: rýchlo vytvorený predchodca prototypu
- prototyp: vytvorenie prototypu aplikácie na základe spätnej väzby
- zjemnenie: prototyp je zjemnený podľa pripomienok používateľa
- údržba: údržba vydanej verzie

Každú fázu ďalej rozložili do modulov, ktoré sú priradené konkrétnym ľuďom. Jeden modul predstavuje prácu o časovej náročnosti v rozsahu do dvoch týždňov pre jednu osobu a je riadený špirálovým modelom vývoja.(**Obr. 2**.)



Obr. 2. Špirálový model vývoja softvéru [5]

Detailný plán bol vytvorený vždy len pre nasledujúcu fázu. Ako východisko slúžili hlavné termíny uzávierok. Na stanovovanie dĺžky trvania modulov použili skúsenosti pracovne starších členov tímu. Tí ohodnotili moduly ako ťažký, stredne ťažký alebo ľahký. Na základe tohto ohodnotenia sa určil čas potrebný na dokončenie modulu. Dodržovali stanovenú maximálnu dobu pre modul. V prípade, že odhad predstavoval dlhší časový úsek, modul bol dekomponovaný.

Použitím takéhoto prístupu zaznamenali zvýšenie presnosti plánov. Boli schopní jednoduchšie a presnejšie odhadnúť koľko bude práca na module trvať. Ukázalo sa, že pri manipulácii s menšími modulmi môže človek zreteľnejšie sledovať postup v práci, a teda lepšie zhodnotiť, či sa práca oneskoruje.

Autori celkovo priaznivo hodnotia skúsenosť s popísaným prístupom, nakoľko sa zlepšila schopnosť plánovania a odhadu.

Softvér pre americké Ministerstvo energetiky

Beams vo svojej práci [1] oboznamuje čitateľa s prístupom k vývoju softvéru označeným ako SPICE (Software Prototyping in Changing Enviroment). Vznikol počas práce na softvéri pre americké ministerstvo energetiky a je určený pre rýchlo sa meniace prostredie.

Meniace sa prostredie v tomto ponímaní znamená to, že požiadavky používateľov sa menia aj v rámci jednotlivých týždňov. Niektoré funkcie softvéru prestanú byť požadované, požiadavky na iné zase vzniknú.

Autori popisujú rozdelenie práce na nasledovné typy úloh.

Krátkodobé projekty

Mali za cieľ riešiť hlavné požiadavky používateľov na základnú funkcionalitu. Výsledkom týchto úloh boli makra, ktoré sa integrovali už do existujúcich softvérových nástrojov. Aby sa používatelia naučili tieto makra používať, bola potrebná užšia spolupráca medzi používateľmi a vývojármi založená na častých stretnutiach. Vďaka nej následne vývojári lepšie pochopili požiadavky používateľa, používatelia sa zase stali zručnejšími pri používaní softvérových nástrojov.

Dlhodobé projekty

Ich úlohou bolo zakomponovať nové rozsiahle vlastnosti. Pre ich úspech bolo potrebné, aby mali nemenné požiadavky. Pracovali na nich vždy noví zamestnanci, ktorí ovládali vždy najmodernejšie technológie. Tí nemali kontakt so zákazníkom.

Model divokej vody

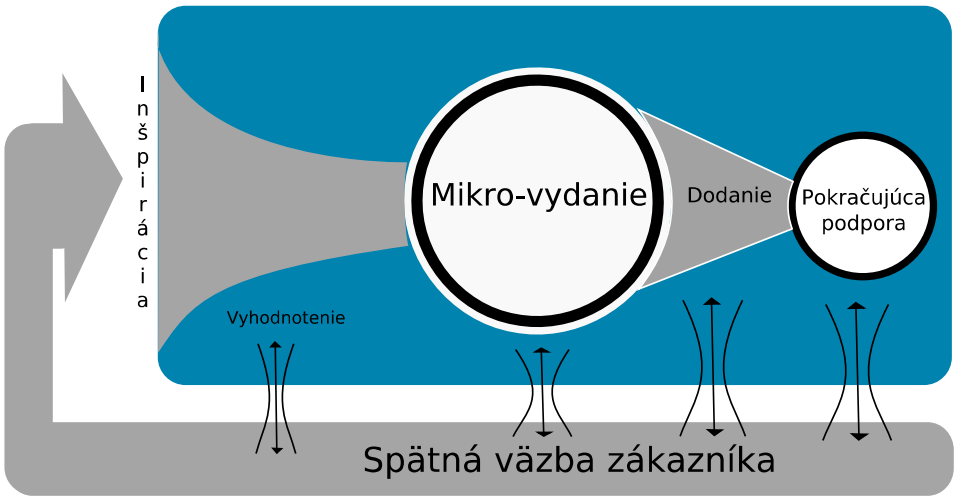
Štúdia [3] vykonaná na troch malých spoločnostiach v Spojených štátoch amerických popisuje model tzv. divokej vody. Autori týmto názvom zvýrazňujú metaforu, ktorú im na prvý pohľad procesy vnútri spoločností pripomínali.

Všetky tieto spoločnosti mali spoločné rysy:

- malý počet zamestnancov
- obmedzená zákaznícka základňa
- dodávka produktov cez Internet
- jedna produktová línia

Analýzou spoločností sa zistilo, že napriek prvotnému obrazu využívajú podobnú štruktúru vývojového procesu.

Jadrom tohto modelu (**Obr. 3**) tú tzv. „mikro-vydania“, ktoré takmer každý týždeň dodávajú nové vlastnosti na trh. Vývojový cyklus je tak skrátený na krátke obdobie. To umožňuje zapracovávať nové vlastnosti do produktu rýchlejšie a demonštrovať postup na práci. Rovnako skrátenie vývojového cyklu umožňuje zníženie rizika. Spoločnosť nemusí investovať financie a čas na analýzu a môže požiť koncept „vyskúšaj a uvidíš“, ktorý môže byť niekedy lacnejším prístupom.



Obr. 3. Model divokej vody [3]

Dimenzie plánu

Časová dimenzia plánu

Ako vo všetkých disciplínach, aj v softvérovom inžinierstve dochádza k posunu v uvažovaní. Pred dvoma desaťročiami bol bežne využívaný vodopádový model vývoja softvéru. V poslednom období, ako aj uvedené práce uvádzajú, sú do vývoja softvéru zavádzané rôzne opakujúce sa cykly s pomerne krátkou dobou trvania. Práve tieto cykly by mohli byť tou hľadanou odpoveďou na otázku: „Na ako dlhé obdobie treba plán vytvárať?“

Je ale takýto prístup vhodný pre všetky možné situácie. Je tým zázračným všeliekom?

Ak vychádzame z predpokladu, že vytvárame softvérový produkt pre zákazníka, ktorý si vytvorenie tohto produktu objednal, je rozumné aplikovať práve spomínaný postup. Z praxe je všeobecne známe, že zákazník nikdy nevie presne, že čo vlastne chce. Je to dané spôsobom ako ľudia uvažujú. Nezamýšľajú sa nad podrobnosťami každého jedného detailu. Často ich zaujíma iba základný všeobecný pohľad. Až neskôr, keď vidia a môžu si časti hotového softvéru vyskúšať, uvedomia si súvislosti, na ktoré skôr nemysleli. Zákazník sa taktiež často nezmieni o základných veciach vo svojej problémovej oblasti, pretože ich považuje za samozrejmé. Práve takéto zdroje zmien sa ukazujú ako kritické pre dodržanie plánu. Niektoré vplyvy ako ochorenie člena vývojového tímu alebo menej závažné vplyvy oneskorovacieho charakteru môžeme ošetriť vložením rezervy do plánu. Vplyvy, ktoré spôsobujú, že

plán prestane byť aktuálny, možno však len ťažko ošetriť nejakými alternatívami plánov, navyše ak sú alternatívy neznáme. Obmedzením plánovania na kratšie obdobie môžeme práve tieto rizika zmeny eliminovať, keďže ešte nebude existovať plán, ktorý by sa mohol stať neaktuálnym.

Ako by to mohlo byť v prípade generického produktu? V tomto prípade by sme mohli skĺzavať k presvedčeniu, že táto situácia je diametrálne odlišná. Domnievam sa ale, že tieto prípady sú značne podobné. V takejto situácií sa práve sami posúvame do pozície zákazníka. Možno je predsa len naše postavenie o čosi lepšie, pretože ako technici sme zvyknutí formulovať naše myšlienky často jasne a exaktne. Taktiež nakoľko vytvárame produkt my sami, sme s ním v neustálom kontakte. Na druhej strane sme ľudia a navyše nemusíme mať z problémovej oblasti tak kvalitné znalosti ako by mal externý objednávateľ. To prináša značné riziko zmien. Z tohto dôvodu som presvedčený, že je lepšie zvoliť kratšie obdobie, pre ktoré sa urobí podrobný plán. Po jeho splnení sa skončení je vhodné verifikovať správnosť vytvoreného produktu, ohodnotiť plán a pustiť sa do ďalšej fázy.

Dimenzia pochopenia činností

Podľa môjho osobného názoru netreba tvorbu plánu vnímať len z pohľadu stanovenia časového harmonogramu. Stanovenie plánu so sebou neprináša len túto jednu dimenziu. Aj keď nám predsa len ako prvá napadne táto časová súvislosť, pri vytváraní plánu získame aj ďalšie vedľajšie produkty, ktoré sa nám môžu zísť.

Aj keď sa rozhodneme vytvoriť podrobný plán, jednou z ďalších výhod, ktoré získame je vytvorenie si lepšieho obrazu o činnostiach, ktoré potrebujeme vykonať. Osobne keď nad niečím uvažujem, snažím sa myslieť viac v súvislostiach, vytvoriť si prehľad a nehľadím vtedy až tak veľmi na jednotlivé podrobnosti. Práve často až v momente, keď sa začnem zamýšľať nad jednotlivými krokmi, ktoré musím vykonať, uvedomím si: „Aha, na toto som nemyslel.“

Z tohto pohľadu môžeme vytváranie podrobného plánu chápať nie ako cieľ ale ako nástroj, ktorý nám pomôže lepšie pochopiť vykonávané úkony. Ako hovorí Paleta v [4], projektový plán sám o sebe nemá byť chápaný ako dogma. V tomto prípade to platí azda ešte väčšmi. Celkovo si myslím, že vytvorenie podrobného plánu má svoje opodstatnenie, časové intencie je však vhodné určiť iba na kratšie časové obdobie a netreba sa obávať plán pozmeniť, ak sa ukáže, že je to potrebné.

Záver

Plánovanie je jednou z činností, ktorú je pri tvorbe softvéru takpovediac nutné vykonať. Jej prevedením sa celý proces tvorby softvéru zlepší. Projekt získa rysy profesionálneho prístupu, nebude to už len slepé blúdenie v tme. Preto si namiesto otázky: „Či plánovať?“ kladieme otázku: „Ako plánovať?“

Táto esej sa preto venuje dvom rozmerom plánovania: dĺžke a podrobnosti plánu. V súvislosti týchto otázok rozoberá vplyv použitého prístupu k vývoju softvéru na tvorbu plánu. Tieto dve dimenzie sú ilustrované na prípadoch z reálnej praxe,

v ktorých autori popisujú svoje skúsenosti s tvorbou softvéru v malých tímoch a v meniacom sa prostredí.

Použitá literatúra

1. BEAMS, J.D., Adding spice to software development: a software development approach designed for rapidly changing environments. In *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*. New York : ACM Press, 1995, s. 384-389.
2. BECK, K., et. al. *Manifesto for Agile Software Development* [online]. 2001 [cit. 2007-10-20]. Dostupné na: <http://www.agilemanifesto.org>
3. HARRIS, M., AEBISCHER, K., KLAUS, T. The whitewater process: software product development in small IT businesses. In *Communications of the ACM*. New York : ACM Press, 2007, roč. 50, č. 5, s. 89-93.
4. PALETA, P. *Co programátory ve škole neučí aneb Softwareové inženýrství v reálné praxi*. 1. vyd. Brno : Computer Press, 2003. ISBN 80-251-0073-1. 360 s.
5. RETTIG, M., SIMONS, G. A project planning and development process for small teams. In *Communications of the ACM*, New York : ACM Press, 1993, roč. 36, č. 10, s. 45-55.

Annotation

Software development approach impact to plan making

Plannig has its own place in software development process. It brings benefits to project scale, cost and duration estimation, task assigning, etc. Nowadays no one denies need for planning. Though every person who is about to create plan asks questions: “How detail should be plan? For what time period should be plan created?“ As a software development approach used in practice influences different software development aspects, it also affects project plan specifics. This essay deals with software development approach impact to planning and analyses listed questions.