# Je možné predvídať budúcnosť? (Plánovanie softvérového projektu)

MAROŠ MAJERČÍK

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
majercik04[zavináč]student[.]fiit[.]stuba[.]sk

**Abstrakt.** Manažment vývoja softvérového projektu je dlhodobý proces, ktorý v sebe zahŕňa niekoľko samostatných fáz, ktoré by mal každý projekt absolvovať. Jednou z najdôležitejších, ak nie vôbec najdôležitejšou fázou tohto procesu, je plánovanie. Práve v tejto počiatočnej fáze je možné projektu zabezpečiť úspešný koniec alebo, v horšom prípade, vopred ho odsúdiť na neúspech. Na nasledujúcich stránkach tohto dokumentu sa pokúsim bližšie priblížiť proces plánovania softvérového projektu. V prvej časti budem hľadať odpovede na často kladené otázky súvisiace s plánovaním softvérového projektu: Prečo strácať drahocenný čas vytváraním plánu? Vyplatí sa vytvárať dva rôzne plány ? Ako je možné vopred naplánovať niečo, o čom nemáme žiadne, alebo len minimálne znalosti ? V druhej časti sa zameriam na jeden z často používaných prístupov v plánovaní projektu, ktorým je iteratívne plánovanie. .

## Úvod

Jedným z najvýraznejších príkladov zlyhania plánovania modernej doby sa objavil 6. júna 1944 presne o 6:30 doobeda na pláži medzi dovtedy neznámymi mestami Vierville a Colleville v severnom Francúzsku. Ak by sme merali hodnotu plánu podľa počtu vecí, ktoré daný plán predvída a kontroluje, tak by sa práve spomenutá časť invázie v Normandii počas druhej svetovej vojny hodnotila určite ako neúspešná.

Keď sa snažíme odhadnúť náklady na projekt a vyprodukovať plán pre jeho úspešné ukončenie, je to skoro to isté , akoby sme sa pokúšali predvídať budúcnosť. To však vôbec nie je jednoduché, o čom svedčí aj výrok uznávaného fyzika a držiteľa Nobelovej ceny Nielsa Bohra, ktorý raz poznamenal: „Predvídanie je veľmi náročné, obzvlášť ak je v ňom zahrnutá budúcnosť.“ Všeobecne je známe, že mnoho projektov nebolo dotiahnutých do úspešného konca práve kvôli nezvládnutej alebo dokonca úplne vypustenej časti plánovania v procese tvorby projektu, či už softvérového alebo iného. Avšak ani dobre vytvorený plán projektu ešte nezaručuje jeho úspešné zakončenie, pretože vývoj softvérového produktu jednoducho nie je exaktná veda.

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

## Prečo plánovať?

Takže prečo sa vôbec zaoberať plánovaním, ak sú všetky doposiaľ uvedené fakty pravdivé? Ak v plánovaní naozaj existuje taká vysoká miera nepredvídateľnosti, ktorú vopred nie je možné odhadnúť, nebolo by jednoduchšie vrhnúť sa do tvorby projektu „po hlave“ a riešiť problémy, až keď sa objavia? Každý, kto mal možnosť pracovať na slabo naplánovanom projekte, vie, že takéto riešenie nie je prijateľné, hlavne pokiaľ sa jedná o projekt väčších rozmerov. Z vlastnej skúsenosti môžem potvrdiť, že hoci je tvorba projektového plánu nepríjemná súčasť projektu (aspoň pre mňa určite je), zároveň je aj veľmi dôležitá. Nepríjemná môže byť ešte viac v prípade, ak robím plán projektu, na ktorom budem osobne pracovať. Hlavne keď vyberám termíny ukončenia jednotlivých častí projektu, o ktorých už v danej chvíli viem, že ich bude potrebné dodržať, pričom v tom momente mám o projekte veľmi chabé informácie.

I napriek faktu, že plánovanie má svoje hranice, mala by táto fáza tvoriť neoddeliteľnú súčasť projektu, dokonca aj v prípade, keď ide o projekt s vysokým stupňom premenlivosti. Každý plán by mal pritom riešiť aspoň tieto kritické časti[3]:

- *Vytýčiť ciele projektu* – tým zaručíme, že každý člen tímu pracujúci na projekte má presnú predstavu o tom, čo sa v projekte snažíme dosiahnuť
- *Pripraviť sa na predpokladané prekážky* – do plánu je potrebné zahrnúť udalosti, o ktorých predpokladáme, že nastanú a zároveň treba poskytnúť akúsi mieru pripravenosti na ich riešenie.
- *Očakávať neočakávané* – v pláne je tiež vhodné identifikovať niektoré neočakávané udalosti, ktoré by sa mohli objaviť a tiež navrhnúť ich riešenie, alebo prinajmenšom aspoň identifikovať, že takáto udalosť nastala.
- *Zabezpečiť zdroje* – dobrý plán v sebe musí zahŕňať základné zdroje, ktoré pokryjú náklady na očakávanú prácu, ako aj primerané množstvo zdrojov na pokrytie neočakávaných udalostí, v závislosti od pravdepodobnosti, že daná udalosť nastane

To, čo plán samozrejme nemôže zabezpečiť, je úplné pokrytie neočakávaných udalostí, ktoré sa môžu (aj keď nemusia) objaviť počas vývoja projektu. Samozrejme, každý plán má takéto obmedzenie, ktoré určuje presnosť celého plánu a tým aj jeho schopnosť predvídať a teda kontrolovať budúcnosť.

## Ako sa vysporiadať s premenlivosťou?

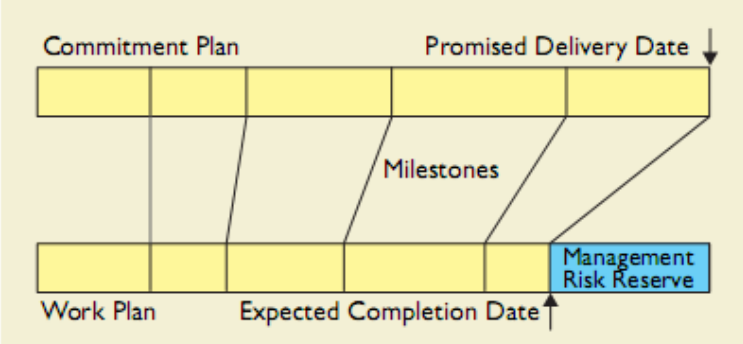
Ďalším kritickým bodom pri tvorbe každého plánu je premenlivosť. Pod týmto pojmov je možné si predstaviť niečo, čo nepoznáme a preto nemôžeme vopred vedieť, ako to ovplyvní samotný projekt. V softvérových projektoch existuje niekoľko zdrojov premenlivosti [3]:

- *Rozsah projektu* – často vopred nie je možné určiť, aký rozsiahly bude výsledný systém, aké funkcie bude vyžadovať a čo nás bude stáť vytvorenie týchto funkcií
- *Dodržanie požiadaviek* – rovnako vopred nemusíme vedieť, ako sa nám bude dariť pri získavaní a zatrieďovaní znalostí potrebných pre vývoj projektu. Existuje veľké množstvo faktorov, ktoré môžu ovplyvniť našu schopnosť dodržať požiadavky zákazníka, pričom plnú kontrolu máme len nad niekoľkými z nich.
- *Technológia* – ovplyvňuje obidva predchádzajúce faktory, pretože nové zmeny v technológii, ktoré bývajú veľmi časté, môžu mať vážny dopad na správanie sa projektu

Každý zo spomenutých faktorov prináša určitý stupeň nepredvídateľnosti do tvorby plánu, čo je samozrejme nežiaduce a preto ich nie je možné ignorovať. Najjednoduchším spôsobom, ako sa vysporiadať s nepredvídateľnými faktormi, je predstierať, že neexistujú. Môžeme jednoducho dúfať, že nepriaznivé udalosti, ktoré by mohli spomaliť vývoj projektu, jednoducho nenastanú. Tiež môžeme dúfať, že ak sa aj nejaké podobné udalosti vyskytnú, budú v rovnováhe s pozitívnymi udalosťami, ktoré sa objavia, a ktoré naopak urýchlia vývoj projektu. Ak by sme však pristúpili na hociktorú z dvoch uvedených možností, nebolo by to nič iné, ako hazardovanie s projektom, preto je potrebné zvoliť sofistikovanejší prístup. V prvom rade je potrebné pripustiť, že nepredvídateľné faktory existujú a preto ich nemôžeme ignorovať. Práve naopak, musíme sa s nimi primerane vysporiadať.

## Dva sú vždy lepšie ako jeden

Jednou z možností, ako riešiť nepredvídateľnosť a premenlivosť projektu, je vytvoriť plány dva. Jeden, ktorý by počítal len s očakávanými rizikami a druhý, v ktorom by boli zahrnuté aj tie neočakávané.




Obr. 1. Koncept dvoch plánov

*Pracovný plán* – v tomto pláne sú zahrnuté zdroje, o ktorých predpokladáme, že budú pre projekt nevyhnutné na základe našich poznatkov (pravdepodobne neúplných) o projekte v čase, keď vytvárame plán. Ide o tradičný typ projektového plánu, podľa ktorého sa riadia členovia tímu pracujúci na projekte.

*Záväzný plán* – tento plán v sebe zahŕňa kompletný pracovný plán, navyše však obsahuje určitú rezervu, ktorá sa určí na základe množstva nejasností, ktoré o pláne máme v čase jeho vytvárania.

Treba poznamenať, že rezerva obsiahnutá v záväznom pláne súvisí hlavne s nejasnosťami, ktoré vyplývajú z nepresnosti informácií, ktoré nám poskytne zákazník pred začatím plánovania. Práve tieto nejasnosti sú hlavným zdrojom rizík v projekte. Samozrejme, projekt tiež môže obsahovať neurčitosti súvisiace s inými faktormi (ako je napríklad rozsah projektu), tie by však mali byť zahrnuté už v pracovnom pláne. Skutočnosť, že zákazníkovi „podstrčíme“ rozdielny (vzdialenejší) termín ukončenia ako ten, ktorý je uvedený v pracovnom pláne, nie je vôbec odlišná od prístupu v ľubovoľnej inej oblasti obchodu, kde zákazníkovi stanovíme vyššiu cenu ako je výrobná cena.

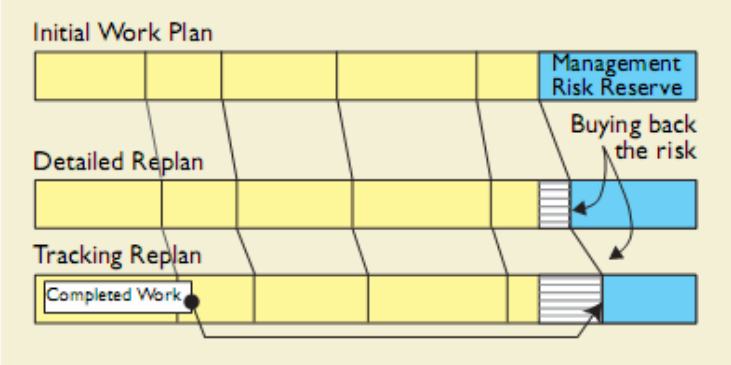
## Okrádame zákazníka?

Na prvý pohľad sa môže zdať, že prístup dvoch plánov, keď zákazníkovi predkladáme iné termíny ako tie, ktoré sa snažíme v skutočnosti dodržať, by mohol byť problémový. Skúsme sa pozrieť na vec z druhej strany, t.j. zo strany zákazníka, ktorému predložíme záväzný plán. Otázka znie, či je rozumné „skrývať“ skutočný dátum odovzdania hotového projektu pred zákazníkom a predstierať, že termín uvedený v záväznom pláne je ten skutočný? Čo ak na to zákazník príde ? Nebude sa cítiť, akoby sme sa ho snažili podviesť, či okradnúť?

Na prvý pohľad sa môže zdať, že zákazníka skutočne „okrádame“ tým, že za projekt požadujeme viac zdrojov, ako by v skutočnosti bolo potrebné. Ako to však väčšinou býva, každá minca má dve strany a nie je tomu inak ani v tomto prípade. Ukázalo sa, že keby sme zákazníkovi oznámili, že projekt ukončíme presne v deň, ktorý sme si stanovili v pracovnom pláne, riziko a následky spojené s nedodržaním termínu by odniesol kompletne buď zákazník alebo (čo je pravdepodobnejšie) vývojársky tím. Takáto vyhliadka samozrejme nie je veľmi priaznivá.

Výhoda dvoch plánov spočíva práve v tom, že vývojársky tím je ochotný zdieľať riziko spojené s nedodržaním termínu spolu so zákazníkom. Práve preto je potrebné vyhradiť určité dodatočné zdroje uvedené v záväznom pláne. Samozrejme tento princíp dvoch plánov bude fungovať len v prípade, že zdroje rezervované navyše budú využívané rozumne. To znamená na pokrytie problémov vzniknutých v dôsledku nejasností v pláne a nie na „ulievanie sa v práci.“ A ak sme plánovali správne, tak hoci riešenie nejasností v priebehu práce na projekte bude mať za následok oneskorenie sa voči pracovnému plánu, tak toto oneskorenie neprekročí termín, ktorý sme stanovili ako záväzný pre zákazníka.

## Zníženie rizika nie je zadarmo




Obr. 2. Využívanie rezervných zdrojov

Na obrázku č. 2 je možné vidieť, čo sa v skutočnosti zvyčajne stane. Prvotný odhad je takmer vždy nepresný. Preto je potrebné vyhradiť rezervné zdroje na základe odhadovaných rizík v projekte. Postupom času, ako sa neskôr dostávame k podrobnejšiemu plánovaniu jednotlivých súčastí, keď sme schopní lepšie porozumieť tomu, čo vlastne máme spraviť, začneme využívať rezervné zdroje podľa potreby. Dalo by sa povedať, že využívame výhodu rezervných zdrojov ako protihodnotu za podstúpenie rizika, ktoré sme boli ochotní na seba vziať kvôli nejednoznačnostiam v projekte. Inými slovami povedané, „nakupujeme za podstúpené riziká“.

Tým, že máme dostatok času na analýzu nejasných súčastí projektu, narastá aj naše pochopenie týchto súčastí a zároveň klesá riziko. Preto je úplne adekvátne, že zákazník musí zaplatiť o niečo viac, ak chce znížiť riziká súvisiace s projektom. Tento princíp funguje aj v bežnom živote, pri nákupe v obchode (napríklad platíme navyše za poskytovanú záruku) a rovnako platí aj pri softvérových projektoch.

## Iteratívne plánovanie

Ako bolo spomínané v predchádzajúcej časti, rozsiahle softvérové projekty sú v mnohých prípadoch ťažko plánovateľné kvôli neurčitostiam, ktoré existujú na začiatku projektu. Jednou z často používaných techník, ako tento problém odstrániť, je iteratívne plánovanie.

Každý úspešný manažér plánovania vie, ako vytvoriť kvalitný plán projektu. Musí vedieť identifikovať a stanoviť kritické body (míľniky) a podľa nich adekvátne naplánovať celý projekt. Viackrát už bolo ukázané, že ak niektorý z týchto kritických bodov prekročíme, je len veľmi ťažké dohnať stratený čas [1]. V takomto prípade môže dôjsť k situácii, kedy bude potrebné natiahnuť pôvodný plán, alebo „okresať“ projekt, tzn. vynechať niektorú z jeho častí.

Ani jedna z uvedených možností určite nie je príliš lákavá, našťastie existuje aj ďalšia alternatíva. Ak vieme rozdeliť projekt na menšie celky a zároveň zabezpečiť dostatočnú pružnosť v projektovom rozvrhu, potom je možné presmerovať kritický bod a dokončiť projekt so všetkými plánovanými súčasťami načas. Tento spôsob vývoja projektu v menších samostatných moduloch, kde každý z modulov prechádza vlastným životným cyklom, sa nazýva iteratívne plánovanie.

## Kedy sa používa?

Najčastejšou chybou pri plánovaní veľkých projektov býva nesprávne určenie míľnikov v projekte na základe nedostatku informácií o funkcionalite, ktorú treba vytvoriť. Tento problém väčšinou nastane v prípade, keď zákazník požaduje ukončenie niektorej fázy projektu v príliš krátkom čase bez toho, aby vedel čo vlastne požaduje od vývojárskeho tímu. Vývojársky tím potom musí robiť rozhodnutia skôr, ako je pripravený dané rozhodnutia vykonať. V takomto prípade je ideálne použiť iteratívne plánovanie, ktoré umožní tímu začať pracovať na projekte v čase, keď sú známe len základné vlastnosti projektu, pričom kritické body sú už napevno určené.

Základný princíp iteratívneho plánovania spočíva v tom, že celý projekt rozdelíme do niekoľkých modulov, ktoré potom riešime samostatne. Každý z modulov prechádza celým životným cyklom, pričom podrobnejší plán vytvárame len pre modul, ktorý je ďalší v poradí. Tento spôsob plánovania je ideálne spojiť s objektovým vývojom projektu, nakoľko obidve techniky (plánovacia a programovacia) majú spoločnú modulárnosť.

## Plánovanie nasledujúcej fázy

Jedna z hlavných výhod skutočnosti, že nie je potrebné urobiť podrobný plán celého projektu, spočíva v tom, že projekt môže postupovať vpred oveľa rýchlejšie. Mnoho vývojárov sa totiž vopred „vystraší“, keď uvidia vytvorený rozsiahly plán s veľkým množstvom úloh. Môže sa im zdať, že nie je možné, aby v takom zložitom projekte spravili nejaký pokrok alebo si pomyslia, že ak pri takom rozsiahlom projekte dôjde k oneskoreniu, nebude to taký veľký problém. Samozrejme takéto myslenie je nesprávne, pretože každé malé oneskorenie v projekte sa výrazne odrazí v záverečnej fáze projektu. Ak sa však vytvorí len plán určitej časti modulu, nepôsobí to tak odradzujúco a zároveň to motivuje členov tímu k lepším výkonom. Postupným pridávaním detailov je potom možné lepšie vidieť pokrok, ktorý bol v projekte vykonaný.

Pri klasickom spôsobe plánovania s pevným rozvrhom dochádza často k plytvaniu času (napríklad pri skoršom ukončení úlohy), a to z niekoľkých dôvodov [1]:

- Nie je dôvod na začínanie úlohy vopred, takže sa na úlohe začne pracovať v poslednej minúte. Podľa Goldratta sa tento prístup nazýva ako tzv. „študentský syndróm“, keď čakáme až od poslednej možnej chvíle, kedy treba na pridelenej úlohe začať pracovať. (Tento názov je podľa mňa viac ako výstižný a určite nebol vybraný náhodne.)
- Keď pracujeme na viacerých úlohách naraz, dochádza k zbytočnému plytvaniu času v dôsledku neustálej zmeny kontextu medzi jednotlivými úlohami, na ktoré sa treba prispôsobovať.
- Zviazanosť medzi jednotlivými úlohami môže mať za následok neefektívne využívanie pracovných síl, ak sa vopred neidentifikuje. Môže totiž dôjsť k situáciám, keď nie je možné pokračovať v postupe jednej úlohy, pretože je priamo závislá na výstupe druhej úlohy, ktorá ešte nebola dokončená.

Pri iteratívnom plánovaní nemá žiadny z členov tímu k dispozícii dlhý zoznam úloh, preto je jednoduchšie sledovať pokrok v práci. Menší počet úloh tiež zvyšuje motiváciu pre začatie práce na nasledujúcej úlohe. Keďže plán je stále viac a viac zjemňovaný počas vývoja, zviazanosť medzi jednotlivými úlohami je možné odhaliť včas. V prípade, že ich objavíme viac naraz, je možné znovu preplánovať daný modul.

## Záver

Podrobne prepracovaný plán vylodenia spojencov v Normadii spomínaný v úvode tohto článku ani zďaleka nepredvídal a ani nekontroloval výsledky vylodenia. V ten deň v skutočnosti takmer nič nefungovalo tak ako malo. Ak sa na vylodenie pozrieme z pohľadu dodržania vytvoreného plánu, tak ho jednoznačne môžeme posúdiť ako neúspešné. Na druhej strane z pohľadu splnených cieľov bolo vylodenie úspešné. Na obranu tvorcov plánu však je možné spomenúť výrok nemeckého generála Helmutha von Moltkeho: “Žiadny bojový plán ešte neprežil stretnutie s nepriateľom“ [3]. Aj keď softvérové projekty väčšinou nie sú až také premenlivé ako bojové pole, v mnohých ohľadoch majú veľa spoločného. S trochou odľahčenia by sa dalo povedať, že „žiadny softvérový plán úplne neprežil stretnutie s projektom“. Samozrejme v skutočnosti to nie je až také, ako to na prvý pohľad vyzerá. Veď úlohou plánu softvérového projektu nie je presne do bodky určiť, akou cestou sa má projekt uberať. Primárnym cieľom plánu je načrtnúť smer a odhadnúť potrebné zdroje tak, aby sa minimalizovalo riziko nesplnenia cieľov projektu.

## Použitá literatúra

1. Johanna Rothman: Iterative software project planning and tracking, URL: http://www.jrothman.com/Papers/7ICSQ97.html, 1997, (24.10.2007).

2. Marc Retting and Gary Simons: A project planning and development process for small teams, *Communications of the ACM*, Vol. 36, No. 10 (1993) october, (24.10.2007).
3. Phillip G Armour: To plan, two plans, *Communications of the ACM*, Vol. 48, No. 9 (2005) september, (24.10.2007).

## Annotation

*Is it possible to predict future? (Software project planning)*

Software project management of project development is a long-run process, which includes several phases that every project should get through. One of the most important phase, maybe the most important, is planning. It is the initial phase where we can secure successful completion of the project or in the worse case scenario to convict it to the failure. In the next few pages I will try to introduce you into the process of software project planning. In the first part I will search for answers which are connected with software project planning: Why bother with planning of the project at all? Is it necessary to create two plans? How it is possible to predict something that we have limited knowledge about? In the second part of the document I will focus on one of the most common approaches in software project planning known as Iterative project planning.