OPTIMIZMUS JE SPÔSOBENÝ NEDOSTATKOM INFORMÁCIÍ, PESIMIZMUS ICH NADBYTKOM

Problémy sa riešia najľahšie čisto zdravým rozumom a bez zbytočného panikárenia.

Vojtech Juhász

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

Abstrakt. *Od malička máme svoje predstavy o svojej budúcnosti, svoje sny a ciele. Na ich dosiahnutie väčšinou potrebujeme si vytvoriť plány a podľa nich sa riadiť. Nie je to inak ani v softvérovom inžinierstve. Na dosiahnutie vytýčených cieľov potrebujeme cieľavedome postupovať podľa dobre premysleného plánu. Táto esej je venovaná plánovaniu. Na začiatku si predstavujeme čo presne máme na mysli keď rozprávame o plánovaní a predstavíme niekoľko nástrojov a metód, bežne využívaných pri vytváraní plánov. Ako v bežnom živote, aj počas vývoja softvéru sa vyskytujú nepredvídané deje ktoré môžu zmeniť, alebo úplne pokaziť naše plány. Uvedieme niekoľko príkladov, aj z vlastných skúseností, na to ako sa prerábajú plány.*

Kľúčové slová: *plánovanie, manažment softvéru*

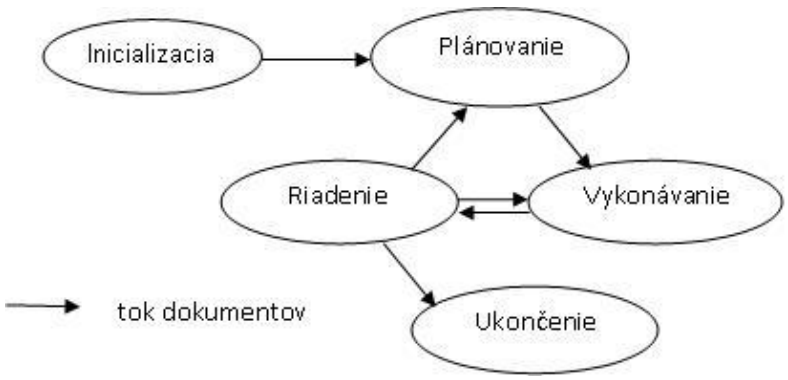
Čo je plánovanie

Všetky projekty sú charakterizovateľné tým, že na nich pracujú ľudia, využívajú ohraničené zdroje, sú plánované, vykonávané a overené [1]. Plánovanie je jedným zo základných manažérskych procesov, bez ktorého sú projekty väčšinou odsúdené na neúspech. Na druhej strane však plánovanie nie je postačujúcou podmienkou úspechu projektu. Plánovanie projektu zastrešuje viacero oblastí ako: náklady, rozsah a rozvrh projektu, ľudské zdroje, komunikácia, riadenia rizík a obstarávanie. Každú z týchto uvedených činností ešte môžeme ďalej deliť napr. plánovanie rozvrhu projektu zahŕňa procesy potrebné na úspešné dokončenie projektu v stanovenom čase, ktoré sú podľa [1]:

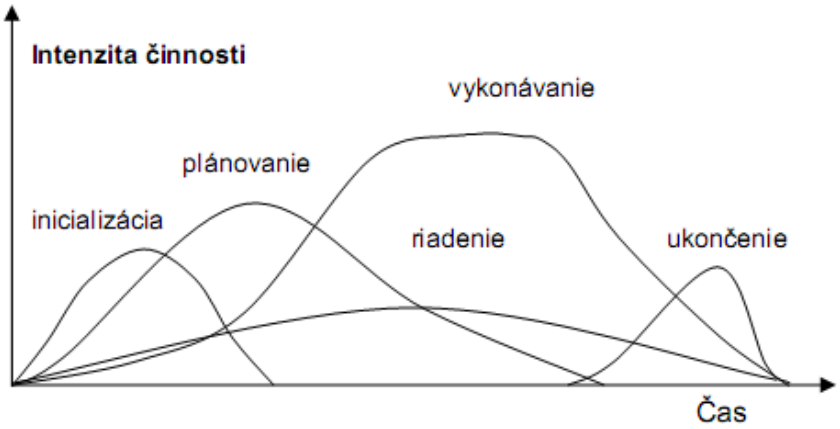
1. Definícia činností – je aktivita, ktorá je potrebná na to, aby sme mohli vyprodukovať rôzne deliverable pre jednotlivé etapy projektu

2. Delenie činností – sú dokumentované rozsahy a nadväznosti jednotlivých činností.
3. Odhad trvania činností – odhaduje sa počet pracovných cyklov potrebných na dokončenie jednotlivých činností
4. Časovanie vývoja – zastrešuje analýzu postupností činností, trvaní činností a potrebných zdrojov na vytvorenie rozvrhu projektu.

Vyššie uvedené procesy sa môžu prekrývať a navzájom sa ovplyvňovať.



Obrázok 1: Plánovanie softvérového produktu [2]



Obrázok 2: Intenzita jednotlivých činností počas vývoja softvéru [2]

Ako je to znázornené aj na obrázku 1, plánovanie nie je jednorazová činnosť ale naopak, je vykonané opakovane počas celého životného cyklu projektu. V jednotlivých fázach sa intenzita plánovania mení: na začiatku je plánovanie intenzívnejšie pričom ako sa približujeme k odovzdaniu projektu, intenzita plánovania klesá [2], ako je to znázornená aj na obrázku 2.

Ak je to možné, plánovanie je vykonané formou hierarchickej dekompozície úloh [3], potom zdroje sú alokované tak, aby sa zaistila produktivita personálu. Je veľmi dôležité aby počas plánovania boli určené a prediskutované možné riziká projektu a to nie len

v manažmente, ale aj so klientmi. Po určení rizík sú určené aj postupy i zodpovednosti na zaistenie kvality výsledného produktu. Spolu s procesmi verifikácie a validácie sú určené aj termíny pre rôzne prehliadky a audity.

Plánovanie umožňuje detekciu slabých častí projektu ešte pred tým že spôsobila problémy a tým pádom manažéri môžu zvažovať a prehodnocovať svoje rozhodnutia, prípadne určité časti projektu, ktoré daný problém môže ovplyvniť.

Po zostavení prvého plánu, manažér plánovania môže odhadnúť cenu všetkých prostriedkov, ľudských a materiálnych zdrojov potrebných na realizáciu projektu.

Prečo projekty meškajú?

Zvyčajne za každým projektom stojí nejaká objednávka, nejaký zákazník. V záujme zákazníka je aby objednaný produkt mu bol čím lacnejšie dodaný v požadovanej kvalite a v stanovenom termíne. Veľakrát sa stane, že projekty meškajú. Odpoveď na otázku *Prečo projekty meškajú?* môže byť veľmi zložitá, ale vo viacerých prípadoch jedným z dôvodov je všadeprítomný a neeliminovateľný ľudský faktor. V [5] autor triedi ľudí do troch skupín:

1. Optimisti – sú ľudia, ktorí si myslia, že nič nie je nemožné a zároveň sú presvedčení aj o tom, že všetko je aj ľahké, práve preto ich odhady sú vo väčšine prípadov nereálne.
2. Pesimisti – sú ľudia, ktorí sú presvedčení o tom, že ak niečo je zvládnuteľné, tak potom je to iba ťažko zvládnuteľné, preto sa do riešenia problémov ani nepustia.
3. Realisti – sú ľudia, ktorí vedia (presne) odhadnúť potrebné úsilie na dosiahnutia určitých cieľov.

V softvérovom inžinierstve na zvládnutie problémov potrebujeme hlavne optimistov a realistov. Prečo? Prečo vôbec potrebujeme optimistov, ak ich odhady sú nepresné? Prečo sa nebavíme o pesimistoch? Na tieto otázky hľadáme odpovede v nasledujúcich odsekoch.

Na ilustrovanie účinkovania uvedených typov ľudí si uvedieme príklad na základe[5]. Predstavme si veľký projekt, ktorý trvá dva roky. Pesimista by odhadol trvanie projektu na tri roky, lebo počíta so všetkými možnými komplikáciami. Optimista však odhadol iba na pol roka. Keď spoločnosť začne realizovať projekt, postupne zistí nedostatky plánu. Rozvrh sa postupne rozrastá, kým po dvoch rokoch je dokončený. Aj keď s veľkým meškaním, ale projekt je dokončený, zákazník možno má výhrady kvôli meškaniu, dodávacia firma možno za sklz musí zaplatiť aj pokutu. Ale keď objednávateľ začne používať daný systém, príde na to, že systém je dobrý a dostal všetko čo si objednal.

V prípade, že realista alebo pesimista pre zákazníka prezentuje projekt s tým, že mu to bude trvať dva alebo tri roky, zákazník si môže vybrať na realizáciu iného, alebo môže ten projekt celkom zrušiť. Hoci realisti sú určite potrební na to aby sme projekt mohli držať pod kontrolou, svojimi reálnymi ponukami majú menšiu šancu zaujať a získať objednávateľov. Ako poučenie si môžeme povedať, že optimistov potrebujeme preto aby sme softvérové projekty mohli začať a realistov potrebujeme na úspešné zvládnutie problémov a dokončenie projektu.

Ako si vytvoríme dobrý plán?

Z vyššie uvedeného príkladu môžeme vidieť, že softvérové projekty veľakrát nemeškajú kvôli problémom, ale často iba kvôli nereálnemu plánu. Plánovanie je náročný proces, nezávisle na tom kto ho vykonáva, optimista, realista alebo pesimista. Keďže si myslím, že extrémom je lepšie sa vyhýbať, zaoberať sa budem s planom zostavením realistami.

Na podporu tejto činnosti sú vyvinuté rôzne nástroje a postupy. Správnosť týchto postupov alebo nástrojov sa posudzuje vždy individuálne, podľa potrieb projektu, podľa vedomostí a skúseností manažéra plánovania. Práve preto sa zameriavam na všeobecné princípy, ktoré je dobré dodržiavať.

Je veľmi dôležité aby bol plán prehľadný a aby sa dal ľahko sledovať. Práve preto najvhodnejšou formou na zachytenie plánu je tabuľka. Alternatíva plánovacej tabuľky je navrhnutá v [6]. V tejto plánovacej tabuľke sa nachádzajú najpodstatnejšie informácie o jednotlivých moduloch a s nimi spojených elementárnych úlohách:

1. Modul projektu (funkcionalita)
2. Elementárna úloha
3. Priorita úlohy
4. Pôvodný odhadovaný čas
5. Aktuálny odhadovaný čas
6. Využitý čas
7. Zostávajúci čas
8. Zodpovedná osoba

V takejto tabuľke môžeme sledovať dekompozíciu jednotlivých modulov na elementárne úlohy pre programátorov. Táto dekompozícia je najťažšou i najdôležitejšou časťou zostavenia plánu [6]. Správne určenie elementárnych úloh je veľmi dôležité aj preto, lebo jeho pomocou sa začne vykresľovať obraz celého modulu: t.j. potrebné kroky (úlohy), algoritmy alebo štruktúry, už počas vytvárania plánu.

Po dekompozícií na elementárne úlohy potrebujeme určiť resp. odhadnúť čas potrebný na realizáciu jednotlivých úloh. Tieto odhady je najlepšie robiť spolu s osobou, resp. programátorom, ktorý bude zodpovedný za danú časť. Takýto spôsob odhadovania času nám pomôže zostaviť čo najpresnejšie plány.

V tabuľke je spravovaný aj odhadnutý aj reálne potrebný čas na úlohy. Dôvodom prečo spravovať pôvodný a aktuálny odhad je jednoduchý: väčšina programátorov nevie odhadnúť ako dlho im bude trvať riešenie danej úlohy [6]. Práve preto je veľmi užitočné keď programátor vidí, aké odhady robil a ako dlho mu daná úloha reálne trvala, aby sa mohol poučiť zo svojich chýb.

Ak chceme vytvoriť reálny plán, potom pri odhadovaní času potrebného na jednotlivé úlohy musíme zohľadniť aj fakt, že každý blok je vhodné otestovať príp. ladiť. Práve preto je potrebné do plánu zahrnúť čas na testovanie, ladenie a integráciu. Občas sa stane, že manažéri podcenia časovú náročnosť testovania i ladenia. Aj z vlastných skúseností viem potvrdiť, že pod tlakom konečných termínov (deadline) programátor občas svojvoľne – aj pod tlakom svojho manažéra – zvolí radšej „prekryť" niektoré problémy, namiesto ich riešenia, aby vedel preukázať funkčnosť riešenia v požadovanom termíne. Vediac túto skutočnosť, môže nás napadnúť otázka: je to správne vytvárať tlak

z manažmentu na podriadených, aby za každú cenu dodržiavali termíny? To určite *nie*. Je dôležité aby každý programátor mal k dispozícii čas na ladenie riešenia svojich úloh. Práve preto je veľmi dôležité aby už pri vytváraní plánov manažéri počítali s časom na ladenie. Takto spôsobené chyby sa môžu veľmi dlho skrývať. Čím neskoršie sa chyby nájdu, tým je drahšia ich oprava. Chybu je najľahšie opraviť ešte vtedy, keď sme danú úlohu urobili. Čím neskôr sa tomu začneme venovať, tým ťažšie to bude.. Veď kto už ladil programy vie, že ladenie je najľahšie vtedy, keď program máme ešte v hlave. Kým celú logiku, ciele ešte vidíme pred svojimi očami. Čím neskoršie začneme hľadať chyby, tým väčšiu šancu máme na to, že zablúdime vo svojich kódoch, nehovoriac o tom, keď chyby potrebujeme hľadať a opravovať po niekom inom. Hľadanie chýb je ako veda: nikdy nevieš kedy sa ti podarí problém vyriešiť [6]. Riešenie aj tých najjednoduchších problémov môže zaberať veľa času ak sa vydáme nevhodným smerom.

Po odstránení chýb z danej úlohy, potrebujeme naše riešenie zintegrovať s ďalšími časťami implementovanej funkcionality. Táto integračná fáza je ďalším potenciálnym zdrojom nejasností počas plánovania. Manažér nemôže presne vedieť, či všetko prebehne úspešne a integrácia bude pozostávať iba napr. z kompilovania spolu s ostatnými modulmi alebo budeme musieť lúštiť záhadné chybové hlášky. Práve preto je potrebné vyhradiť dostatočný čas aj na integráciu jednotlivých modulov.

Na softvérových projektoch pracujú ľudia, občas si musia aj odpočinúť troška. Práve preto je potrebné aby sa pri zostavovaní plánu rátalo s výpadkami ako dovolenky alebo práce neschopnosti jednotlivých členov realizačného tímu. Mnohokrát sa stane, že manažéri nepočítajú s takýmito výpadkami. Každý vie, že oddýchnutý pracovník, hlavne ak ma aj dobrú náladu a úžitok z toho na čom pracuje, pracuje efektívnejšie a rýchlejšie ako vyťažený zamestnanec pracujúci s nechuťou. Práve preto je nevyhnutné hlavne ak ide o plán na niekoľko mesiacov alebo rokov, zahrnúť aj dovolenky pre jednotlivých pracovníkov. Plán bez zohľadnenia tejto skutočnosti môže byť nereálny a meškanie môže postupne narastať ako sa zvyšuje vyčerpanosť a nespokojnosť vývojárov. Ak zamestnanec je nespokojný, môže sa rozhodnúť za radikálne úkony, ako zmena zamestnania alebo štrajk. Tieto fakty by znamenali ďalšiu záťaž pre plán vo forme zaškolenia iného zamestnanca, príp. hľadaním nového odborníka. Kým sa rozbehne nový zamestnanec na danom projekte, ten predchádzajúci sa môže vrátiť z dovolenky a pokračovať vo svojej činnosti.

Ako manažér ovplyvňuje realizovateľnosť plánu

Občas si manažéri myslia, že tesné konečne termíny (tight deadline) budú motivovať programátorov aby pracovali rýchlejšie [6]. Neuvedomujú si, že ľudia pod tlakom sa väčšinou cítia depresívne a akoby odsúdení na neúspech. V takomto duševnom stave programátori sú menej produktívni, teda takýto postup manažéra je určite iba na úkor projektu. Práve preto, manažéri by mali určovať míľniky spolu s programátormi, resp. po konzultáciách s nimi, ako sme to uvádzali vyššie.

Ak manažér vidí, že nie je možné dodržať termíny podľa plánu, má na výber medzi niekoľkými možnosťami: buď bude programátorov preťažovať, čo sme už spomínali že väčšinou nie je vhodné, alebo sa môže rozhodnúť vynechať určité plánované súčiastky alebo môže sa rozhodnúť, že prijme nových programátorov.

Pridávanie nových pracovníkov na projekt väčšinou nie je dobrým riešením. Prečo? Podľa [7] pridávanie nových pracovníkov na projekt, prácu ešte viac zdržuje. Dôvodom toho sú hlavne neznalosti prostredia a projektu, počiatočné komunikačné ťažkosti. Na vyriešenie týchto ťažkostí, manažér musí vynaložiť extra úsilie na rozbehnutie nových členov. Tieto úsilia môžu ľahko presahovať prínosy nového zamestnanca, čo sa určite odzrkadľuje aj v pomere človeko-hodín strávených užitočnou prácou na projekte a tých čo sme potrebovali na *opatrovanie* nového člena tímu. Pridanie človeka na projekt na jeden mesiac automaticky ešte neznižuje dĺžku projektu o jeden človeko-mesiac.

Ak nechceme aby odovzdanie projektu bolo (príliš) oneskorené, manažér sa musí rozhodnúť o vynechaní alebo odložení niektorých úloh do nasledujúcej verzie. Ako vybrať úlohy na odstránenie? Predovšetkým potrebujeme zvažovať prínos jednotlivých úloh k celkovému výsledku. Je dôležité aby jednotlivé úlohy boli zoradené podľa priority. Tým sa môžeme vyhnúť situáciám, keď programátori vyberú zábavnejšiu ale menej prínosnú časť programu na implementáciu, hoci má menšiu priradenú hodnotu.

Prečo projekty zlyhajú?

Na otázku nemáme jednoznačnú odpoveď, ale odvážim sa povedať, že najčastejšími dôvodmi sú rôzne chyby pri zostavovaní plánov. V [8] autor sa zaoberá najčastejšími chybami, ku ktorým sa dochádza pri plánovaní:

1. Neplánovať vôbec
2. Neplánovať všetky aspekty projektu
3. Neplánovať riziká
4. Používanie toho istého plánu pre všetky projekty
5. Používanie vopred daného plánu na všetky projekty
6. Vytvorenie detailného plánu dopredu
7. Plánovanie že meškanie si dobehneme neskôr
8. Dovolenie divergencie projektu od plánu
9. Nepoučiť sa z predchádzajúcich chýb

Počas štúdia sme riešili rôzne školské projekty, nejaké riešime aj teraz i v budúcnosti pravdepodobne budeme riešiť projekty, možno aj väčších rozsahov. Tí ktorí sme tomu neverili, na vlastnej koži sme zažili zlyhanie rôznych projektov a teda aj ich dôsledky – opakovanie predmetov ☺. Dôsledky zlyhania reálnych projektov môžu byť a pravidelne asi aj sú o mnoho vážnejšie. Dôvody zlyhania? Zlý plán a všetko čo sa z neho vyplýva: nedodržanie termínov, odovzdanie projektov bez testovania, odovzdanie s vynechaním rôznych funkcionalít.

Väčšinu takých problémov môžeme obísť zostavením dobrého plánu. Tie projekty ktoré žiadané plánovanie nepoužívajú ani by sme nemali nazývať projektmi, veď softvérový projekt podľa definície [10] je „(...) činnosť s definovanými cieľmi, rozpočtom a plánom (...)“.

Keď si predstavíme že vytvoríme podrobný plán na nasledujúce dva tri roky, už aj intuitívne musíme vedieť, že 6. nemá zmysel. Keďže žijeme v neustále sa meniacom prostredí nevieme predpovedať alebo odhadnúť všetky zmeny, ktoré majú vplyv na náš projekt. Práve preto je potrebné vynaložiť úsilie radšej na zostavenie hrubého plánu a na

jeho postupné zjemňovanie. Tým sme sa dostali k problému 3. Či chceme alebo nechceme, nemôžeme si strčiť hlavu do piesku a tváriť sa, že zmeny sa nás nedotýkajú. Práve naopak, musíme byť pripravení na všetky možné neplánované ale predvídateľné situácie. S tým súvisí aj sledovanie zmien a prispôsobenie plánu aktuálnej situácii. Ak plán nesleduje meniace sa okolnosti, môžeme sa ocitnúť v situácii keď sa realita bude podobať plánu iba do malej miery, čo znamená ďalší problém v plánovaní ako je to uvedené aj v 8.

Keď nejaká firma má úspešný produkt vyvíjaný na základe nejakého návrhu postupujúc podľa nejakého plánu a dostane ďalšie zákazky na vytvorenie podobného produktu môže sa stať, že manažéri si zostavujú plán podľa toho úspešného produktu. To nie je problém, kým zohľadňujú špecifické požiadavky daného projektu.

Jednou častou chybou, ktorej som sa dopustil často bolo, že som svoje úlohy odkladal s tým, že po určitom čase už budem vedieť viac podrobnosti o danej úlohe a svoju úlohu vyriešim ako na dotyk čarovným prútikom. Samozrejme že sa to nestalo. Hoci som mal teoretické vedomosti, chýbali mi praktické skúsenosti, ktoré by som mohol získať počas času kým som úlohu odkladal. Naučil som sa, že 7. nie je dobrou cestou.

Každý človek môže robiť chyby, každý manažér sa môže pomýliť pri zostavení plánu, urobiť nesprávne rozhodnutie. Najväčšou chybou čo môžeme urobiť po plánovaní, je nepočítať s problémami na ktoré sme narazili pri riešení niektorých z našich predchádzajúcich projektov. Je veľmi dôležité, aby sme si postupne zhromaždili aj skúsenosti popri osvojených vedomostiach, aby sme sa naučili ktorou cestou máme kráčať aby sme sa v meniacom sa prostredí úspešne mohli riešiť rôzne úlohy, resp. projekty.

Vlastnosti dobrého manažéra plánovania

Na základe doteraz spomenutých tém si môžeme načrtnúť obraz dobrého manažéra plánovania. Dobrý manažér plánovania odstraňuje prekážky a zabezpečuje zdroje na zrealizovanie projektu [9]. Odstránením prekážok a zabezpečovaním zdrojov by mal začať ešte pred začatím ďalších prác na projekte. Dobrý manažér problémy odstraňuje ešte pred ich objavením práve návrhom dobrého a jasne definovaného plánu.

Zodpovedný manažér plánovania pri plánovaní by mal zistiť i určitým spôsobom zohľadniť aj odhady projektového tímu, ktorý celú prácu bude vykonávať. Aj tak sa môže stať, že z nejakých dôvodov sa projekt bude meškať o mesiac, dva alebo viac.

Ideálny manažér by nemal iba ovládať oblasť plánovania projektu ale tiež by mal udržiavať plán aktuálny. Je potrebné aby vedel usúdiť ktoré časti projektu nesú v sebe takú pridanú hodnotu, bez ktorej by sa hodnota, resp. cena klesla na neprijateľnú úroveň a tiež by mal určiť tie hodnoty, ktoré sú menej cenné. V [6] plán porovnávajú k celku poskladaného z drevených blokov, ktoré je možné pridávať alebo ju odoberať. V krízových situáciách manažér musí zvážiť, že čo môže stratiť alebo získať implementáciou jednotlivých komponentov a na základe tohto sa rozhodnúť o prioritách projektového tímu a o tom čo má z projektu vynechať resp. ešte dorobiť. Teda manažér musí rozhodovať ktoré drevené bloky môžeme odstrániť a ktoré potrebujeme mať v danom celku.

Pri plánovaní nemôžeme podľahnúť panike a začať robiť bez rozumného rozhodnutia: skracovať čas na testovanie, tlačiť na poriadených na miesto ich motivovaním, a bez rozumne pridávať ľudí na projekt. Pridávaním ďalších človekohodín

priradením nových ľudí ešte neznamená okamžité zníženie času potrebného na implementáciu projektu [7].

Záver

J. Billings raz povedal, že: *„Genialita nie je nič iné, iba zdravý sedliacky rozum oblečený do slávnostných šiat"*. Toto tvrdenie môže platiť aj ohľadom plánovania softvérového projektu. K zostaveniu dobrého plánu potrebujeme rôzne informácie o mnohých veciach, ako dostupnosť zdrojov, schopnosti implementačného tímu, ale tie informácie nikdy netvoria dobrý plán, bez manažéra so správnym myslením, bez manažéra ktorý zohľadňuje všetko možného a to iba logikou t.j. zdravím tzv. sedliackym rozumom. Schopnosť robiť dobré úsudky na základe dostupných informácií je vlastnosť ktorý rozlišuje dobrého manažéra plánovania od ostatných manažérov.

Použitá literatúra:

1. Project Managment Institute: A guide to the Project Managment Body of Knowledge. 3rd Ed., 2000
2. Bieliková, M: Softvérové inžinierstvo. Princípy a manažment. 1. vyd. Bratislava: STU, 2000. 220 s.
3. Abran, A., Moore, J.W.: Guide to the Software Engineering Body of Knowledge. Dostupné na internete: http://www.computer.org/portal/web/swebok/htmlformat [cit.: 2010-10-12]
4. Churchville, D.: Small teams make better software. Dostupné na internete: http://www.extremeplanner.com/blog/2006/08/small-teams-make-better-software.html [cit: 2010-10-10]
5. Kesteloot, L.: Why software is late. Dostupné na internete: http://www.teamten.com/lawrence/writings/late_software.html [cit: 2010-10-10]
6. Spolsky J.: Painless software Schedules, Dostupné na internete: http://www.joelonsoftware.com/articles/fog0000000245.html [cit: 2010-10-10]
7. Brooks, F.P.: The mythical man-month: essays on software engineering. Addison-Wesley Professional, 2004.
8. Mc Connel, S.: The nine deadly sins of project planning. In IEEE Software, Vol. 18, No. 5, 2001, 5-7.
9. Retting, M., Simons, G.: A project planning and development process for small teams. In Communications of the ACM, Vol. 36, No 10, 1993, 45-55.

Annotation

We have been imagining our future since our childhood, we have our dreams and targets. We need to create plans and follow them to achieve our targets. This is the same in software engineering. To reach our set targets we need to advance alongside well premeditated plans. This essay is dedicated to planning. At the beginning we point out what we understand by planning and introduce few

means and methods used for composing plans. Just like in everyday life, even in the path of software development may occur some unforeseen occurrences which can slightly or completely change the original plan. I show several examples how to deal with the changing of plan, adding some examples from my own experiences, too.