METÓDY SKRÁTENIA HARMONOGRAMU POČAS REALIZÁCIE PROJEKTU

Projekt bez kritickej cesty je ako loď bez kormidla.

D. Meyer

Jaroslav Prokop

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
jaro.prokop@gmail.com

Abstrakt. *Pri plnení projektového plánu vznikajú vždy situácie, kedy sa plnenie niektorých kľúčových úloh oneskorí a pre to je potrebné skrátenie následných úloh tak, aby boli dodržané dôležité termíny. Napriek existencii rôznych metód skrátenia sa väčšinou toto skrátenie realizuje len na základe výberu manažérov projektu. V eseji uvádzam existujúce metódy skrátenia plánu projektu, ich výhody a nevýhody. Výsledkom je zistenie, že aplikovanie metód na skrátenie je z hľadiska určenia relevantných atribútov pre všetky úlohy extrémne náročné. Navyše nie je možné určiť tieto atribúty presne a to môže viesť k nie optimálnemu riešeniu. Kvôli týmto dôvodom sa drvivá väčšina metód na skrátenie v praxi nevyužíva. V práci sa zamýšľam nad dôvodmi doterajšieho neúspechu metód skrátenia a nad možnosťami ich použitia v reálnom projekte tak, aby priniesli požadovaný výsledok. Porovnávam metódy založené na princípoch kritickej cesty a jednoduchého ohodnotenia úloh s metódami založenými na zložitejších grafových metódach so komplikovaným ohodnotením všetkých hrán, teda nie len úloh, ale aj vzťahov medzi nimi. Výsledkom skúmania tejto práce je, že pri riešení praktických problémov na rozsiahlych projektoch je dôležité, aby boli metódy čo najľahšie použiteľné bez nutnosti získavania veľkého množstva dodatočných informácií.*

Kľúčové slová: *skrátenie harmonogram plán riadenie plánovanie časové rezervy*

Úvod

Už dlhú dobu sa plánovaniu prikladá veľká váha. S rozšírením počítačových a informačných technológií sa plánovanie posunulo na vyššiu úroveň. Sú dostupné mnohé nástroje a pribúdajú taktiež ďalšie a ďalšie metódy zaoberajúce sa určitou časťou plánovania. Cieľom tejto úvahy sú problémy spojené s metódami zameranými na skracovanie plánu. Skracovanie je potrebné vždy keď sa nejaká úloha oneskorí a ohrozí nedodržanie koncových termínov projektu, alebo ak sa z nejakého dôvodu posunie koncový termín projektu na skorší termín a k nemu potrebujeme prispôsobiť projektový plán, respektíve harmonogram. Ak nastane takáto situácia, máme k dispozícii mnoho rôznych metód, v závislosti od toho aký spôsob plánovania používame na to, aby sme si správne vybrali úlohu, alebo úlohy, ktorých plánovaný termín a trvanie skrátime. V reálnom živote však zistíme, že to nie je také jednoduché. Väčšina metód potrebuje k správnemu vyhodnoteniu aj parametre, s ktorými sa zväčša v praxi nepočíta. Tu vyvstáva problém ktorému sa budem v eseji podrobnejšie venovať.

Plánovanie je možné v rôznych oblastiach, ja sa v nasledujúcich riadkoch zamyslím nad plánovaním rozsiahlych softvérových projektov. V rozsiahlych projektoch, akými sú napríklad softvérové projekty pre štátne, či iné veľké inštitúcie, alebo výstavba zložitých technologických zariadení, pri plánovaní vypláva na povrch množstvo problémov, s ktorými sa môžeme stretnúť aj pri plánovaní menších, či väčších softvérových projektov. Najlepší príklad poskytujú veľké projekty a to kvôli rozsahu i kvôli všemožným situáciám, ktoré v nich môžu nastať a v praxi aj nastávajú.

Pri úvahách budem vychádzať nie len z uvedenej literatúry a odborných článkov, ktoré obsahujú uvedené metódy a postupy, ale aj z vlastnej skúsenosti, keďže som bol členom tímu zaoberajúcim sa implementáciou rozsiahleho informačného systému pre verejnú správu a neskôr som sa stal priamo členom plánovacieho tímu na dvoch veľkých projektoch v oblasti hutníctva. Všetky tieto projekty mali rozsah rádovo tisícov úloh, ktoré zabezpečovalo viacero dodávateľov (outsourcing), v prípade technologických projektov desiatky dodávateľov. Na týchto projektoch som sa stretol priamo, či nepriamo so skúmaným problémom skracovania harmonogramu.

Nasledujúcu úvahu som rozdelil na tri časti. V prvej rozoberiem základné postupy pri tvorbe plánu čo sa týka vkladania časových rezerv. Tieto rezervy totiž slúžia na to aby sme v prípade sklzu mali z čoho skracovať. V druhej časti potom uvediem reprezentatívne metódy skracovania harmonogramu a zamyslím sa nad atribútmi, ktoré vyžadujú. V poslednej tretej časti sa zamyslím nad použiteľnosťou metód skrátenia harmonogramu vzhľadom na zložitosť ich nasadenia.

Tvorba projektového plánu v praxi

Ak sa vytvára rozsiahly projektový plán, sú do neho vložené okrem jednotlivých úloh a zdrojov aj dôležité míľniky. Tie môžu súvisieť s finančným plnením, nasadením obmedzených kritických zdrojov a nakoniec sú to koncové termíny jednotlivých častí ako aj celého projektu. V prípade softvérového projektu, ktorý používa vodopádový model vývoja, sa jedná väčšinou o míľniky ukončenia analýzy, ukončenia návrhu, respektíve jeho hlavných častí a začatia implementácie, ukončenia implementácie a začiatok testovania,

nasadenia systému a tak ďalej. Je kľúčové, aby boli hlavné míľniky dodržané a najdôležitejší je samozrejme míľnik nasadenia a spustenia systému, respektíve odovzdania systému užívateľovi. Aby to bolo možné, vkladajú sa do plánu časové rezervy pre prípad očakávaného, alebo aj neočakávaného sklzu. Pri tvorbe projektového plánu je vzhľadom na vkladanie časových rezerv možné použiť v princípe tri rôzne prístupy.

Prvý z prístupov by som nazval „veľkorysé plánovanie“, podľa opisu uvedeného v [1] a [3]. V pláne vieme vždy nájsť úlohy, pokiaľ je správne zostavený, ktoré sú rizikové v tom zmysle, že môže ľahko dôjsť k ich sklzu. Napríklad sa môže jednať o úlohy spojené s použitím novej technológie v rámci implementácie, kde treba rátať s problémami, ktoré sú spôsobené jednoducho tým, že táto technológia ešte nebola v praxi odskúšaná. Samozrejme, z rovnakého dôvodu môže dôjsť k sklzu aj v oblasti testovania, alebo nasadenia systému. Týmto úlohám môžeme priradiť viac zdrojov, dlhšiu dobu trvania a takto vytvoriť rezervu pre prípadný problém. Samozrejme treba počítať s tým, či si takéto rezervy môžeme dovoliť vzhľadom na obmedzenia zdrojov, ale aj času. Navyše v softvérových projektoch nie je ničím výnimočným, ak niektorí programátori, návrhári a testovači pracujú súčasne na viacerých projektoch. To musíme pri tvorbe rezerv tiež brať do úvahy. Takto vytvorené rezervy môžeme v prípade potreby zužitkovať na riešenie problému. To znamená že naplánujeme pre každú úlohu, potencionálne hroziacu sklzom, viac času, než je odhad jej trvania. Vnoríme do nej časovú rezervu. Ak budeme kumulovať časové rezervy vo všetkých úlohách, pri prípadnom sklze nejakej úlohy budeme môcť skrátiť časové rezervy nasledujúcich úloh. Nevýhodou tejto metódy je, že takto skryté časové rezervy nie sú dobre viditeľné a sledované úlohy by sa v skutočnosti mali stihnúť skôr ako je uvedené v ich pláne. Pre to som tento prístup nazval veľkorysým plánovaním, pretože v skutočnosti dáme riešiteľom viac času na plnenie úloh, než by mali potrebovať. Toto je veľký problém. Ak sa pozrie manažér do plánu a vidí že sa plán plní na 100% a dosiaľ žiadna úloha neskĺzla, v skutočnosti sme vyčerpali všetky predošlé časové rezervy na všetkých splnených úlohách. Úlohy sa teda neplnili v najlepšom možnom čase, lenže tento problém je ukrytý vo vnorených časových rezervách. Napriek tomuto nedostatku je táto metóda často využívaná, niekedy aj v kombinácii s metódami skracovania harmonogramu uvedenými v nasledujúcej kapitole. Podľa môjho názoru je táto metóda veľmi nebezpečná. Je to kvôli tomu, že keď sa na projekte zúčastňujú viaceré tímy, každý z nich si vypracuje svoj interný plán práce a zdrojov. Keď mu predložíme náš veľkorysý projektový plán s vnorenými časovými rezervami, tím sa mu prispôsobí a však on už časové rezervy neuvidí a berie danú dobu trvania úloh ako smerodajnú. Naplánuje si všetky potrebné zdroje aj čas na obdobie ktoré zahŕňa už aj rezervu. Týmto spôsobom spotrebuje rezervu v podobe času skôr než dôjde k akémukoľvek problému. Potom ak dôjde k oneskoreniu, nemáme už k dispozícii rezervu ktorá na to bola určená. Projekt takto stráca tempo.

V softvérovom projekte má problém vnorených časových rezerv dve roviny. V jednej rovine je plánovanie zdrojov subdodávateľa, ak hovoríme napríklad o outsourcingu programátorského tímu, respektíve o rozdelení realizácie medzi viac tímov, alebo oddelení. V druhej rovine je reálna výkonnosť jednotlivých členov tímu a nadväznosť ich úloh. Nadväznosť úloh je veľmi často aj medzi členmi rôznych tímov, či oddelení, riešiacich ten istý projekt. Hlavne ak jeden tím rieši návrh, iný tím ho implementuje a oddelený môže byť aj tím zaoberajúcim sa testovaním. Oddelením tímov myslím ich

diverzifikáciu v rovine samostatného plánovania zdrojov, ale aj času. Musíme si uvedomiť, ako prebieha plánovanie úloh v softvérovom projekte využívajúcom už spomenutý vodopádový model [1]. Základom plánu je určenie hlavných častí projektu, napríklad analýza, návrh, implementácia, testovanie, nasadenie a následne odhad ich časovej náročnosti [2]. Tieto „hlavné časti" sa ale v skutočnosti dekomponujú až na elementárne úlohy, ktoré rieši konkrétny človek. Takáto dekompozícia zväčša nie je uvedená priamo v hlavnom pláne softvérového projektu, ale je niekedy vyčlenená buď do samostatných detailných harmonogramov, alebo sa tieto úlohy sledujú v kompozícii s vytváraným zdrojovým kódom (príkladom je VSTS 2008). Či už sa jedná o návrh konkrétneho modulu systému, implementáciu grafického rozhrania konkrétneho formulára, alebo otestovanie určitej funkcionality, všetko sú to detailné úlohy. Tu navyše vidíme aj interakciu úloh. Implementácia môže prebehnúť až po návrhu a až následne môže začať testovanie. Problém je teda nasledovný. Ak návrhár nedodá svoj vytvorený model a prípadne ani testovač nepožiada o žiadnu opravu chýb, môže nastať situácia, že programátor nebude mať pridelenú žiadnu úlohu. Tento zdroj v podobe programátora (alebo iného člena tímu) ostane určitý čas nevyužitý. Nie len že tento čas stojí finančné prostriedky, ale tento čas sa ukrajuje z časovej rezervy plánu a môže ju prečerpať. Ako teda súvisí uvedený problém s problémom ktorý som označil ako „veľkorysé plánovanie"? Súvislosť uvedených problémov je v skreslení proporcionality hlavných úloh. Ak pridáme časové rezervy do úloh návrhu a do úloh implementácie, vzájomné proporcie trvania sa oproti pôvodnému odhadu (bez rezerv) zmenia. To potom môže viesť k chybnému plánovaniu zdrojov, keď napríklad vyčleníme viac ľudských zdrojov na implementáciu a menej na návrh. Jedným z výsledkov môže byť nevyťaženosť jedného zdroja a preťaženosť druhého.

Ďalšou metódou je vkladanie časových rezerv do plánu vo forme úloh [2]. To znamená že v reťazci úloh bude tvoriť jedna úloha časovú rezervu a v prípade potreby túto časovú rezervu zužitkujeme. Myslím si že táto metóda je z hľadiska prehľadnosti oveľa lepšia ako predošlá uvedená a však podľa môjho názoru je na väčšom projekte ťažko samostatne použiteľná. Je to pre to, že postupnosti súvisiacich úloh sú väčšinou rôzne previazané a na niektorých miestach môže byť pre to umiestnenie časovej rezervy nevhodné. Ak si predstavíme previazanosť úloh na najnižšej úrovni v softvérovom projekte, môže byť správne určenie väzieb v takom rozsahu priam nereálne. Je pre to podľa mňa vhodné skombinovať tento prístup s nejakou ďalšou metódou.

Mohlo by sa zdať, že by bolo vhodné skombinovať predošlé dve uvedené metódy takým spôsobom, že by sme vyčlenili rezervy z jednotlivých úloh akoby do samostatných úloh. V takom prípade by bola za každou úlohou vložená osobitne ešte aj časová rezerva. Toto riešenie by som však úplne zavrhol. Prečo? Podľa môjho názoru na to existuje hneď niekoľko dôvodov. Za prvé, ak by sme za každú úlohu vložili časovú rezervu, strácal by sa nám význam kritickej cesty, keďže na kritickej ceste by ležalo veľa časových rezerv. Týchto rezerv by sme sa nevedeli zbaviť ani vyfiltrovaním, ak by sme chceli vidieť len skutočné úlohy. Bolo by to pre to, že ak by sme vyfiltrovali časové rezervy, spolu s nimi by sa nám stratili aj s nimi spojené väzby. V konečnom dôsledku by sme potom dostali reťazec úloh, medzi ktorými by boli medzery. Jedna úloha by sa skončila v termíne t a nasledujúca by sa začala až v termíne $t + n$, kde n je dĺžka rezervy, ktorú sme vyfiltrovali. Toto je v príkrom rozpore s princípom kritickej cesty. Kritická cesta nemôže obsahovať takéto časové medzery. Základným problémom by bolo, že ak by sme si zobrazili veľký projekt,

obsahujúci rádovo tisíce úloh, obsahoval by navyše rádovo tisíce časových rezerv. V takomto stave by bol harmonogram značne neprehľadný. Filtrácia by bola, vzhľadom na väzby a medzery medzi úlohami, komplikovaná a nepriniesla by požadovaný výsledok. Keď si navyše predstavíme, že programátori čakajú na zahájenie implementácie už dokončeného návrhu pre to, že sa ich úloha ešte podľa plánu nemá začať (čo je spôsobené vloženou časovou rezervou, ktorú ale nikto nerealizuje a teda ani neskracuje), zásadná nevýhoda tejto metódy je viac než jasná.

Často využívanou metódou vkladania rezerv je časová rezerva na konci projektu [1][4], to znamená že vo vnútri projektu sledujeme plán so skorším koncom ako vykazujeme navonok, pred zákazníkom. Samozrejme, ak si odmyslíme ideálny prípad, tento skorší koniec sa bude postupom projektu čím ďalej tým viac posúvať smerom k skutočnému koncu a v reálnom svete vývoja softvéru ho nezriedka aj prekročí. Podľa môjho názoru je toto najlepšia metóda, pretože každý člen projektového tímu jasne vidí stav plnenia plánu, sklzy v pláne a aj presné odhady termínov bez akýchkoľvek rezerv. I keď každý vie, že na konci je časová rezerva, tak každý má svoje záväzné termíny pre svoje úlohy ktoré sa musí snažiť za každú cenu dodržať. Ak sa mu to ale nepodarí, manažment a plánovač má na konci projektu dostupnú časovú rezervu, aby ňou vykryl vzniknutý sklz. Ak nemá, musí skrátiť niektoré zostávajúce úlohy. Tieto vlastnosti sú poľa mňa veľmi dobrým dôvodom na nasadenie tejto metódy.

Nech použijeme ktorúkoľvek z uvedených metód, pravdepodobne nastane počas trvania projektu situácia, kedy sa kvôli oneskorenej úlohe posunie buď koncový termín projektu, alebo dôležitý míľnik, ako sa uvádza v [4] a [5]. V nasledujúcej kapitole uvediem aké existujú možnosti na úpravu harmonogramu tak, aby sa míľniky projektu vrátili na požadované termíny.

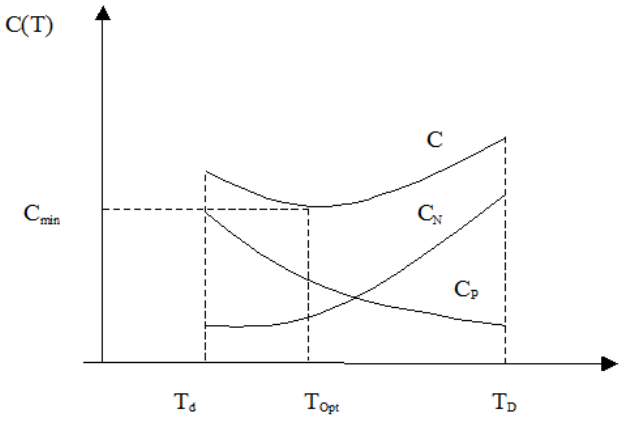
Čomu sa nevyhneme – úprava projektové plánu

Ako som predznamenal v predošlej kapitole, sklz niektorých úloh je nevyhnutný aj keď ho dopredu nie vždy môžeme predpovedať. Ak je tento sklz natoľko podstatný, že ovplyvňuje dôležité termíny či míľniky, nie je iná možnosť ako skrátiť nadväzujúce úlohy. Vzhľadom na použitý prístup pri vkladaní časových rezerv to nemusia byť len úlohy, ale môžu to byť aj rezervy. To však nie je podstatné pre žiadnu z metód na skrátenie plánu, pretože ak je časová rezerva vyčlenená do úlohy, je vnímaná metódou skrátenia len ako obyčajná úloha, zväčša s minimálnou, alebo žiadnou cenou skrátenia.

Metód na skrátenie plánu, respektíve harmonogramu, je veľa. Ich delenie nie je jednoznačné, ale podľa môjho názoru by sa dali rozdeliť na dve skupiny. Prvú skupinu predstavujú metódy založené na kritickej ceste harmonogramu. Kritická cesta je reťazec úloh, ktoré nám v danom momente určujú koncový termín projektu. Druhá skupina potom zastrešuje metódy založené na zložitejších grafových analýzach a algoritmoch. V skutočnosti majú obe skupiny rovnaký cieľ a však líšia sa v tom aké zložité problémy dokážu riešiť a aké algoritmy sú v nich použité.

Ako prvú uvediem metódu CPM/COST podrobne rozobranú v [3]. Je založená na metóde kritickej cesty. Princíp je v ohodnotení činností. Vychádza sa z myšlienky, že náklady na nejakú činnosť závisia od dĺžky jej trvania. Ilustračný príklad môže byť nasledovný. Majme skupinu n programátorov, ktorí implementujú svoje úlohy za čas t.

Náklady, ktoré nás zaujímajú zahrňujú mzdu programátorov, náklady na ostatné zdroje a podobne. A práve tu sa niekedy dá dosiahnuť úspora, ak realizujeme činnosť kratší čas. Ak máme medzi pracovníkmi expertov, tí majú často mnoho násobne vyššiu mzdu ako ostatní. Ak zvýšime počet ostatných pracovníkov tak, aby bola činnosť vykonaná rýchlejšie, budeme expertných pracovníkov potrebovať kratšie. Expertný pracovníci sú v tomto prípade obmedzeným, finančne nákladným zdrojom. Ak ich potrebujeme kratší čas, dosiahneme úsporu. Na druhej strane ale budeme mať väčšiu réžiu kvôli zvýšeniu počtu pracovníkov. Ak ale máme programátorov, pre ktorých návrhári nestíhajú modelovať potrebné diagramy, úsporu určite nedosiahneme. Z toho vidíme, že kratší čas realizácie určitej časti projektu, v tomto prípade implementácie, nie vždy znamená nižšie náklady. V skutočnosti sa s kratším časom zvyšujú iné náklady. Priebeh nákladov vzhľadom na čas je nelineárny. Na nasledujúcom obrázku, obrázok 1 prebraný z [3], je znázornená závislosť nákladov od doby trvania.

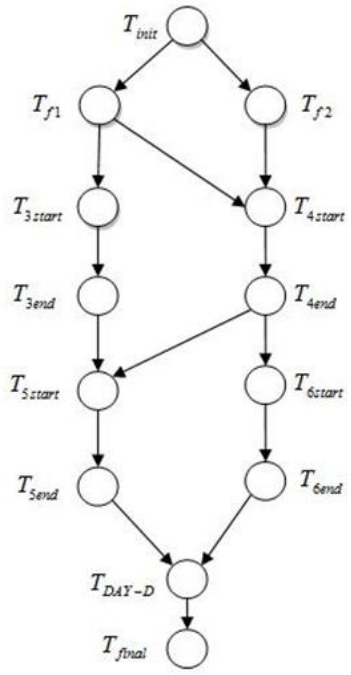


Obr. 1. Závislosť nákladov a času na nejakú činnosť [3]

Krivka C predstavuje celkové náklady na činnosť, krivka C_n predstavuje nepriame náklady a C_p priame náklady. Na zvislej osi sú znázornené náklady a na vodorovnej osi čas. Vidíme, že priebeh nákladov nie je lineárny. Optimálne z hľadiska nákladov je dosiahnuť na krivke C bod $[T_{Opt}, C_{min}]$, ktorý je lokálnym minimom grafu a z hľadiska nákladov je pre nás optimálny. Metóda COST/CPM využíva na výpočet ohodnotenie hrán, ktorými sú jednotlivé činnosti. Pri skracovaní sa hľadá úloha na kritickej ceste, ktorá má pri potrebnej dobe trvania najnižšie náklady, teda najnižšie lokálne minimum. Toto skracovanie sa robí v iteráciách po jednotke času. Je to pre to, že ak skrátime nejakú úlohu na kritickej ceste o jednotku času, môže sa na kritickú cestu dostať ďalšia úloha.

Druhou metódou určenou na skrátenie harmonogramu je metóda skrátenia harmonogramu pomocou minimálneho rezu grafu publikovaná v [6]. Graf zostavíme tak [6], že hrany budú predstavovať aj úlohy, ale aj závislosti medzi nimi. Uzly potom predstavujú začiatok, alebo koniec úloh. To nám umožní ohodnotiť nie len cenu skrátenia úloh, ale aj skrátenia časových rezerv vo forme voľného času medzi dvoma úlohami spojenými závislosťou. Takýmto spôsobom dostaneme podobný graf ako je znázornený na obrázku 2. Jednotlivé vrcholy T predstavujú začiatočné, respektíve koncové body úloh

(T_{nStart}, T_{nEnd}). Body T_{init} a T_{final} reprezentujú začiatok a koniec celého plánu. Bod $T_{Day\text{-}D}$ predstavuje termín ku ktorému sledujeme koncový termín vo vnútri projektu, T_{final} prezentuje koniec projektu navonok, rozdiel medzi týmito dvoma bodmi je časová rezerva. Takto vytvorený graf je na obrázku 2 prebraného z [6]. Táto metóda používa na zistenie úloh, ktoré máme skrátiť algoritmus minimálneho rezu grafu, ktorý sme vytvorili na základe harmonogramu. Metóda minimálneho rezu grafu je v matematike grafov často používaná a známa, ale v tejto metóde bola aplikovaná na harmonogram. Minimálny rez bude prechádzať úlohami, ktoré máme skrátiť tak, aby boli náklady čo najnižšie, čo bolo publikované v [6].



Obr. 2. Grafová reprezentácia harmonogramu/plánu

Pri porovnaní týchto dvoch metód som dospel k záveru, že napriek značne odlišnej implementácii sú si dosť podobné čo sa týka výsledku. Obe metódy vyžadujú dodatočné atribúty. Metóda založená na minimálnom reze grafu samozrejme počíta s väčším množstvom vstupov, akým je napríklad ohodnotenie ceny za skrátenie. Jej nevýhoda je v zložitejšej implementácii. To sa možno na prvý pohľad nejaví ako nevýhoda a však pri metódach riadenia je dôležité, aby boli ich princípy ľahko pochopiteľné. Ak totiž projektový manažér bude chcieť vedieť prečo mu táto metóda vybrala dané úlohy na skrátenie, vysvetlenie algoritmu minimálneho rezu grafu vytvoreného na základe projektového plánu nebude vôbec jednoduché. Naproti tomu princíp skracovania úloh na kritickej ceste je každému jasný. Z tohto dôvodu sa pri nasadení na reálnom projekte prikláňam k metóde CPM/COST.

Implementácia metód skrátenia harmonogramu je v softvérových projektoch zriedkavá a priamo som sa s ňou nestretol. Z nadobudnutých skúseností ale dedukujem

jeden záver. Podľa mňa by sa použitie týchto metód malo realizovať na vyššej plánovacej úrovni, teda na úrovni hlavného projektového plánu. Zahrnúť do skracovaného harmonogramu všetky úlohy jednotlivých členov, teda napríklad jednotlivých návrhárov, analytikov, či programátorov, by bolo neefektívne a v konečnom dôsledku by neprinieslo požadovaný výsledok, ktorým je v skutočnosti skrátenie hlavného harmonogramu a posunutie míľnikov. Po významnom skrátení hlavného harmonogramu, vzhľadom na dĺžku projektu, bude v každom prípade potrebné prehodnotiť a upraviť všetky detailné harmonogramy ktorých sa to dotýka. Napríklad ak skrátime určitú plánovanú etapu návrhu, musíme prehodnotiť nie len dĺžky jednotlivých úloh, ale určite aj ľudské zdroje a pravdepodobne i prioritu a postupnosť jednotlivých úloh tak, aby bolo možné dosiahnuť požadovaný termín.

Existujú aj ďalšie metódy, sú však podobné predošlým uvedeným. Ich základom je vytvorenie určitého grafu a následne použitím rôznych algoritmov a metód na určenie hrán, alebo vrcholov, ktoré tieto metódy skrátia.

V tejto kapitole som čerpal najmä z [3] a [6]

Pre čo sa teda tieto metódy používajú v praxi tak málo?

Toto je otázka ktorú si môže položiť ktorýkoľvek manažér, alebo plánovač. Ja som si ju taktiež položil pri implementácii metódy skrátenia harmonogramu projektu pomocou minimálneho rezu grafu aj v rámci bakalárskej práce a odpoveď bola hneď naporúdzi. Všetky metódy vyžadujú nejaké dodatočné atribúty. Buď len k úlohám, alebo aj k väzbám medzi úlohami. Takým atribútom je napríklad cena za skrátenie, možnosť pridania ďalších zdrojov a podobne. Tu je však kameň úrazu. Nie len z vlastnej skúsenosti, ale aj z poznatkov uvedených v [5] viem, že je náročné správne odhadnúť aj tú najzákladnejšiu veličinu pri plánovaní - dĺžku trvania ľubovoľnej úlohy a to aj pre skúseného odborníka. Dôvodom je, že túto veličinu ovplyvňuje veľa faktorov, ktoré sa väčšinou nedajú odhadnúť dopredu s dostatočným predstihom. Je to napríklad otázka, ktorý dodávateľ bude úlohu vykonávať a či má dostatok zdrojov. S týmto faktorom som sa stretával najčastejšie. Sú aj mnohé ďalšie faktory a táto problematika úzko súvisí s riadením rizík. Tak či onak, vidíme, že odhad základných parametrov jednotlivých úloh je náročný ,často aj nepresný ale hlavne časovo a personálne drahý. Odborníkov, ktorý sú schopný takéto odhady spraviť je veľmi málo. Väčšinou ich na projekte nie je dosť. Aby sme použili metódy na skrátenie, potrebovali by sme spraviť niekoľko násobne väčší počet odhadov rôznych veličín. Jedná sa však o veľmi zložité parametre, pri niektorých úlohách by sme ich jednoducho ani nemohli určiť, alebo len veľmi nepresne čo by viedlo k nepresnému výsledku. V ideálnom prípade by to neboli jednoduché parametre, ale funkcie závislostí nákladov, alebo zdrojov na čase. Keď si predstavíme projekt s hlavným harmonogramom obsahujúcim rádovo tisíce úloh, jedná sa podľa môjho názoru o nerealizovateľnú myšlienku. Pre to sa prikláňam k takému riešeniu, ktoré vyžaduje minimálne navýšenie sledovaných veličín, či už pri tvorbe, alebo pri sledovaní plánu, i keď nebude matematicky najpresnejšie. Myslím si že pri veľkých časovo obmedzených harmonogramoch je dôležitejšia schopnosť rýchlej implementácie z hľadiska tvorby harmonogramu a rýchle dodanie výsledku. Jedinou otázkou je, do akej miery je získaný výsledok správny. To je však na posúdení plánovačov a manažérov daného projektu, pretože v konečnom

dôsledku rozhodujú vždy oni. Nemodifikované metódy, o ktorých som zmienil sa podľa mojej mienky hodia na veľmi malé harmonogramy, alebo na riešenie matematických problémov.

Kvôli uvedeným dôvodom sa prikláňam k použitiu takej metódy, ktorá by brala do ohľadu len akúsi prioritu úloh. Tá by bola určená na základe odhadu zložitosti danej úlohy, rizika jej predĺženia oproti plánu a navýšenia nákladov pri jej skrátení. Na projekte by mala byť vypracovaná metodika určovania tejto priority na základe uvedených veličín tak, aby bol výsledok jediné číslo, jediná veličina. Napríklad, čím by bolo vyššie, tým by bola úloha prioritnejšia a pri skracovaní by sme sa jej snažili vyhnúť. Samotné skracovanie by taktiež podľa mňa nemalo byť úplne automatizované, aby nedošlo k skrytému skráteniu úlohy, ktorú v skutočnosti skrátiť nechceme.

Záver

Napriek dlhej tradícii plánovania a existencii veľkého množstva rôznych metód sú určité oblasti plánovania v praxi menej rozvinuté. Jednou z nich je oblasť skracovania harmonogramov berúc do úvahy náklady, respektíve cenu za skrátenie. Je to zapríčinené veľkou zložitosťou tohto problému a s tým spojenou neochotou venovať toľko času na nasadenie metód, ktoré by ho riešili. Navyše ich nasadenie potom vyvolá sled ďalších problémov spojených s tým, ako tieto metódy efektívne použiť. Napriek tomu, že jednotlivé metódy dokázali, že môžu znížiť náklady na projekt, v teoretickej rovine, v praktickej rovine svoju efektivitu veľmi nepreukazujú. To ma privádza k záveru, že riešenie tohto problému nie je založené na automatizácii skracovania harmonogramu, ale na vytvorení komplexnejších metód a metodík.

Použitá literatúra

1. Lewis J. P.: *Project planning, Schedulinh and Control*. Amacon, 1998
2. Dolanský V., Měkota V., Němec V.: *Projektový management*. Grada, 1996
3. Fiala P.*: Projektové řízení-modely, metody, analýzy*. Praha, 2004
4. Philips V., Teresa L.: *Software Project Management*. 2006
5. Gareis R.: *Happy Projects!* Vienna, 2005
6. Lekavý M., Návrat P.: *Extension of Rescheduling Based on Minimal Graph Cut.* Springer Berlin / Heidelberg, 2008

Annotation

Schedule shortening methods during project realization

There are many situations during project realization, when some key task is delayed. Then it is necessary to shorten next task to reach important milestones and deadlines. In project management, there are many schedule shortening methods. However they are not used often by project management. In essay, I will show existing methods and experiences of project manager during large projects. The result of essay is that is too hard to use existing methods for schedule shortening.

It is because of many attributes, for every single task, are needed. In addition these attributes are also difficult to obtain. Methods for schedule shortening are not used often because of these reasons. I take a thing about how to use these methods effectively. I will also compare critical path methods with graph based methods in this essay. The result of this essay is that simpler methods take advantage against complicated graph methods because of their clumsiness in large project plans.