Sú plány na to, aby sa porušovali?

MARKO DIVÉKY

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
markod@nextra.sk

Abstrakt. Softvérové projekty často končia neúspešne, zlyhajú alebo sú zrušené. Týka sa to projektov všetkých rozmerov – od malých zadaní vytváraných študentmi až po veľké projekty s medzinárodnou účasťou. Takzvaná „softvérová kríza“ aj v dnešnej dobe stále pretrváva, pretože sa doposiaľ nenašlo univerzálne riešenie, ktoré by raz a navždy zabránil zlyhaniu softvéru. Príčin je viacero. Táto esej sleduje aké „zásluhy“ má práve tvorba plánov a plánovanie ako také na neúspešnosti softvérových projektov všetkých rozmerov, objasňuje dôležitosť správneho plánovania, vysvetľuje dôvody, prečo aj dokonalé plány zlyhávajú a v neposlednom rade poskytuje čitateľovi praktické rady ako vytvoriť čo najrobustnejší „nepriestrelný“ plán (nielen) softvérového projektu.

Prežívame krízu...

Softvérové projekty často končia neúspešne, zlyhajú alebo sú zrušené. Platí to od čias obdobia softvérovej krízy, kedy softvér prekračoval svoj finančný rozpočet a časový harmonogram, nespĺňal všetky dohodnuté požiadavky a bol ťažko udržiavateľný. Odvtedy bolo vymyslených viacero metodológií a procesov, ktorých úlohou je predísť takýmto situáciám a zmierniť (dnes už v menšom rozsahu) prežívajúcu krízu. Brooks [3] už dávnejšie predpovedal, že neexistuje univerzálne riešenie predstavujúce akýsi „strieborný náboj“ (angl. „silver bullet“), ktorý vyrieši naraz všetky problémy v softvérovom inžinierstve. Súčasnosť dáva Brooksovi za pravdu, pretože vývoj softvéru je naďalej komplexná a neprebádaná záležitosť a i v dnešnej dobe existujú softvérové projekty, ktoré zažívajú svoju „softvérovú krízu.“

V „softvérovej sieni hanby“ (angl. „Software Hall of Shame“) prezentovanou v časopise IEEE Spectrum [5] ako neúspešné figurujú aj projekty známych nadnárodných spoločností, napríklad Ford, Hewlett-Packard, AT&T Wireless, McDonald’s, Nike a Hilton Hotels. S neúspešnými softvérovými projektmi majú svoje skúsenosti dokonca aj burzy v Londýne či Toronte. Každý jeden z takýchto projektov mal za následok rozsiahlu finančnú stratu, v niektorých prípadoch dokonca problémový softvér priviedol do krachu viaceré firmy.

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

Čia je to vina?

Kto (alebo presnejšie: čo) má teda na svedomí neúspešnosť softvérového projektu? Podozrivých „vinníkov“ je viacero a sú rôzni pre každý softvérový projekt. Všeobecne sa dá povedať, že najčastejšie príčiny problémového softvéru sú nasledovné [5]:

- nereálne alebo nejasné ciele,
- nepresný odhad požadovaných zdrojov,
- zle definované požiadavky,
- slabé monitorovanie stavu projektu,
- nesprávne identifikované riziká,
- chabá komunikácia medzi zákazníkmi, vývojármi a používateľmi,
- použitie nezrelej technológie,
- neschopnosť čeliť komplexnosti projektu,
- neporiadne vývojové praktiky a postupy,
- slabý manažment projektu.

Treba však podotknúť, že softvérový projekt je málokedy neúspešný kvôli práve jednej uvedenej príčine. Väčšinou za neúspechom stojí kombinácia viacerých faktorov a okolností.

Vinník – zlé plánovanie

Zrejme to nie je na prvý pohľad viditeľné, ale podľa môjho názoru veľa spomenutých príčin zlyhania softvérových projektov úzko súvisí s plánovaním. Pozrime sa teda ešte raz na niektoré najčastejšie príčiny, pričom pri každej z nich uvádzam ako je daná príčina previazaná s plánovaním:

- **Nepresný odhad požadovaných zdrojov.** Aj keď odhad požadovaných zdrojov úzko súvisí skôr s požiadavkami na projekt ako s plánovaním, istý súvis tu predsa je. Týka sa to napríklad ľudských zdrojov, ktoré veľmi úzko súvisia s plánovaním, pretože na základe vytvoreného plánu a z termínov v ňom uvedených vyplývajú ľudské zdroje potrebné na dodržanie týchto termínov. Je teda zrejmé, že ak je zle spravený plán, tak aj potrebné ľudské zdroje na projekt budú zle odhadnuté, a to buď naddimenzované (lepší prípad, ale málo častý) alebo poddimenzované (horší, priam katastrofálny prípad, často vedie ku kolapsu celého projektu).
- **Slabé monitorovanie stavu projektu.** Ak si na začiatku určíme nepresný a nekonzistentný plán s kontrolnými bodmi, ktorých miera splnenia sa ťažko

meria, tak aj monitorovanie stavu projektu je tým pádom náročné, nepresné, a teda sa stáva slabým.

- **Nesprávne identifikované riziká.** Takáto situácia ľahko nastane ak identifikácia rizík vychádza z nepresného plánu projektu. Inými slovami povedané, ak je projektový plán zlý, tak riziká identifikované na základe neho budú tiež zlé, resp. nepresné. Platí to aj vice-versa, teda z nepresne identifikovaných rizík vychádza aj nepresný, zlý plán.
- **Neschopnosť čeliť komplexnosti projektu.** S veľkosťou softvérového projektu úmerne rastie aj jeho komplexnosť. Dobrý a kvalitný plán je jedným zo základných predpokladov pre zvládnutie aj toho najväčšieho a najkomplexnejšieho projektu, a teda ak máme nekvalitný a nepresný plán, tak komplexný projekt sa v momente stane nezvládateľným.
- **Slabý manažment projektu.** Príčin vedúcich k slabému manažmentu softvérového projektu je viacero, no jednou z nich je určite aj nepresné a neporiadne plánovanie.

Polovica príčin stojacich za neúspechom softvéru má teda prepojenie a súvis s plánovaním a tvorbou plánov, a preto plánovanie tvorí nutný (ale nie postačujúci) predpoklad k úspechu softvérových projektov. Majúc tento fakt na mysli, poďme sa teraz venovať plánovaniu, jeho významu nielen v softvérovom inžinierstve, príčinám zlyhania plánov, a taktiež veciam, ktorým sa treba pri plánovaní vyhnúť a veciam, ktoré treba brať do úvahy ak chceme aby naše plány boli čo najlepšie.

Je plánovanie dôležité?

Áno! A nielen v softvérovom inžinierstve. Možno si to neuvedomujeme, ale takmer *každá* činnosť, ktorú vykonávame, je istým spôsobom naplánovaná. Napríklad cesta do školy, do práce alebo hocikde inde je plán – musíme si rozmyslieť, či pôjdeme bicyklom, autom, električkou, autobusom, trolejbusom alebo peši. Podľa našej voľby musíme zvoliť aj trasu – môžeme ísť najkratšou cestou, prestupovať alebo okľukou, aby sme sa vyhli rannej zápche. Ďalším príkladom na plánovanie v praxi je nakupovanie, kedy si dopredu spíšeme, čo treba kúpiť a podľa toho zoznamu postupujeme, alebo nakupujeme všetko čo vidíme – aj to je istým spôsobom plán. Dokonca aj táto esej bola napísaná na základe plánu sumarizujúcom o čom budem písať, ktorý som si dopredu vytvoril a spracoval. Podobných príkladov z každodenného života je neúrekom.

Vráťme sa teraz späť k softvérovému inžinierstvu. V predchádzajúcej časti som uviedol tvrdenie, že dobrý plán je základom úspešného projektu. Podobný názor vyslovili vo svojej práci aj Berntsson-Svensson a Aurum [2], ktorí ho podporili ich empirickým pozorovaním. Dôležitosť plánovania a plánov potvrdzuje aj Armour [1], ktorý uvádza štyri dôležité funkcie plánovania:

- **Zverejňovanie projektových cieľov.** Plánovanie pomáha zabezpečiť to, že ľudia pracujúci na danom projekte zdieľajú spoločné porozumenie toho, čo sa projekt snaží dosiahnuť.

- **Príprava na očakávané udalosti.** Plány sa snažia pripraviť celý tím na udalosti, ktoré sa v procese riešenia projektu očakávajú, a poskytujú aj mechanizmy, ako sa s takýmito udalosťami vysporiadať.
- **Predvídanie neočakávaných udalostí.** Dobre vytvorené plány identifikujú niektoré z možných neočakávaných udalostí, ktoré môžu nastať počas riešenia projektu, a poskytujú mechanizmy, ako sa aj s takýmito udalosťami vysporiadať alebo prinajmenšom aspoň ako odhaliť, že udalosti takéhoto charakteru nastali.
- **Rozdeľovanie prostriedkov.** Kompletný plán poskytuje rozdelenie dostupných prostriedkov na prostriedky pre podporu naplánovanej práce a očakávaných udalostí, prostriedky pre podporu adekvátnej časti predvídaných neočakávaných udalostí, a prípadne aj na prostriedky pre podporu nepredvídaných udalostí.

Napriek všetkým výhodám, ktoré nám plány prinášajú, nie je možné nijakým plánom zaručiť úspech projektu [1]. Je to z toho dôvodu, že pri plánovaní a tvorbe plánov skúšame istým spôsobom predpovedať budúcnosť – a to nie je triviálna záležitosť. Plány takisto nevedia úplne „ochrániť“ projekty pred nepredvídateľnými udalosťami – vedia iba vyčleniť prostriedky pre riešenie takýchto situácií, no konkrétne situácie identifikovať nevedia, pretože budúcnosť (zatiaľ) predpovedať nevieme.

Aj dokonalé plány zlyhávajú...

Nesnažím sa tvrdiť a ani vytvoriť dojem, že ak máme vytvorený „dokonalý“ plán pre náš softvérový projekt, tak sme takpovediac „v suchu“ a náš projekt preto nič neohrozí. Aj ten najdokonalejší plán dokáže byť „zrazený na kolená“ nepredvídateľnosťou prostredia reálneho sveta [1]. Alfou a omegou tohto tvrdenia je fakt, ktorý som spomenul vyššie – fakt, že plány sú založené na predpovedaní budúcnosti a budúcnosť je nepredvídateľná. Čím menej vieme o problémovej doméne, do ktorej patrí náš softvérový projekt, tým je väčšia šanca, že počas jeho riešenia sa nám prihodí niečo nepredvídateľné – napríklad neskoro zistíme, že sme zle navrhli architektúru celého projektu. V takejto situácií nám nepomôže ani celá kopa plánov.

Povedzme, že vieme o problémovej doméne nášho projektu všetko a predpokladajme, že počas riešenia projektu nás nič nepredvídateľné nepostihne. Projekt ale mešká a nevieme ho dokončiť v termíne, ktorý stanovuje náš plán – a nikto netuší prečo. Kde sme urobili chybu? Odpoveď je jednoduchá – stačí si uvedomiť skutočnosť, že plány sú vytvárané ľuďmi a tých môžeme rozdeliť na *optimistov*, *realistov* a *pesimistov*. A náš projekt bol naplánovaný práve optimistom. Myslel síce na všetko, pozorne naplánoval všetky dôležité činnosti, ale neuvedomil si, že ich časové trvania zle odhadol skrz jeho prirodzený optimizmus.

S týmto tvrdením súvisí aj zaujímavé pozorovanie Kesteloota [7], ktorý tvrdí, že softvérové projekty meškajú preto, lebo jediní ľudia, ktorí ich naštartujú a plánujú, sú optimisti a tí podvedome podceňujú čas, ktorý bude projekt vyžadovať. Kesteloot tak-

tiež tvrdí, že jemu známe úspešné projekty boli, naopak, plánované realistami a tí sa s veľkou pravdepodobnosťou do rozsiahlych projektov nepustia. Kesteloot preto radí, aby softvérové projekty boli plánované striktne len realistami.

Dovolím si s názorom pána Kesteloota nesúhlasiť. Podľa môjho názoru plány nemusia byť spravidla tvorené výhradne realistami. Jediný problém optimistu ako tvoriteľa plánu je, že nevie adekvátne odhadnúť, koľko budú trvať jednotlivé činnosti na projekte. Tento problém sa ale dá vyriešiť tak, že daný optimista musí nájsť svoju „bulharskú konštantu," ktorou bude násobiť časové obdobia v ním vytvorených plánoch. Tým sa z optimistu prakticky stane realista a projektový tím sa už nemusí jeho plánov obávať... alebo nie? Ja osobne som optimista (zistil som to na základe vlastných skúseností s plánovaním) a momentálne som v procese skúšania svojej vlastnej „bulharskej konštanty."

Zaujímavý názor na dôvod, prečo softvérové projekty meškajú, napísal Brooks vo svojej svetoznámej knihe *The Mythical Man-Month* [4]. Podľa Brooksa za nedodržaním plánov stoja jednak *veľké* príčiny (ktoré sú jednoducho postrehnuteľné a identifikovateľné, ich riešeniu sa ihneď venuje manažment a aplikujú sa patričné nápravné opatrenia), ale aj *malé* príčiny, napríklad choroba člena tímu a následná nemožnosť zvolania porady, výpadok elektriny vo firme, snežná kalamita a ďalšie. Takéto príčiny omeškania sú nepostrehnuteľné a spôsobujú, že projekt mešká „jedným dňom za druhým."

„Recept" na úspešný plán

Predtým, než sa pustíme do konkrétnych rád ako vytvoriť čo najlepší plán si povedzme najprv zopár všeobecných, no účinných rád, ktoré treba mať na pamäti pri plánovaní ako takom. Tieto rady sa na prvý pohľad zdajú byť zrejmé, ale mnohokrát sa na ne pri tvorbe plánov zabúda.

Tím „správnych" ľudí

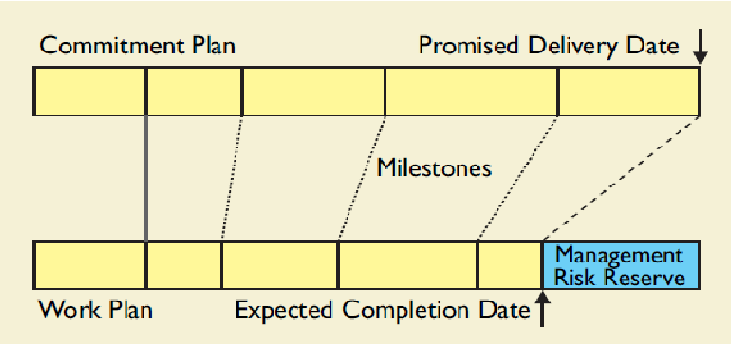
Základným predpokladom pre úspešný plán (a aj úspešný projekt) sú odhodlaní ľudia. Nezáleží na tom, či projekt využíva vodopádový model vývoja softvéru, extrémne programovanie alebo Scrum (či iné metodológie agilného vývoja softvéru) – sú to stále ľudia, kto tvorí jadro projektu.

Aj malý tím správnych a odhodlaných ľudí pokorí tú najprepracovanejšiu metodológiu, nástroj či filozofiu. Na druhej strane, veľký tím nezohraných ľudí nedokáže vytvoriť dobrý výsledok bez ohľadu na to, aké úžasné metodológie či technologické postupy boli v rámci projektu využité [6].

002: Jeden plán nestačí

Ako som už spomenul v predchádzajúcej kapitole, s plánovaním súvisí istá dávka rizika, nakoľko každý plán sa snaží predpovedať (viac či menej neistú) budúcnosť. Taktiež som spomenul fakt, že plány nevedia odvrátiť nepredvídateľné situácie. Zvíťaziť nad budúcnosťou sa (v dobe písania tejto eseje) bohužiaľ nedá, ale dá sa ju aspoň mierne

„preľstiť“. Trik spočíva v tom, že namiesto jedného plánu vytvoríme rovno plány dva – *plán práce* a *záväzný plán* [1], viď Obr. 1.



Obr. 1. Záväzný plán (hore) a plán práce (dole) [1].

Plán práce odzrkadľuje odhadnutý priebeh projektu na základe našich aktuálnych odhadov v dobe vytvorenia plánu. Predstavuje štandardný plán softvérového projektu, podľa ktorého sa riadi celý tím.

Záväzný plán predstavuje plán dohodnutý so zákazníkom, respektíve zadávateľom projektu. Zahŕňa v sebe vyššie opísaný plán práce, no navyše aj dodatočnú rezervu času (a ďalších prostriedkov) pre prípad, že nastanú nepredvídateľné udalosti.

S pomocou dvoch plánov sme očividne viac chránení pred neočakávanými udalosťami než v prípade existencie jediného, a to spoločného plánu. S výhodami ale idú aj nevýhody. Použitie dvoch plánov nesie v sebe riziko rozhnevania členov tímu v prípade, že objavia existenciu „miernejšieho,“ teda záväzného plánu. Jeho „odtajnenie“ členom tímu, na druhej strane, so sebou nesie riziko, že sa všetky naplánované rezervy svojvoľne vyčerpajú – nakoľko z mojej osobnej skúsenosti viem, že väčšina ľudí má tendenciu (neúmyselne) vyplniť všetok dostupný čas i prostriedky.

Čo „najostrejšie“ míľniky

Každý dobrý projekt obsahuje míľniky s presne definovanými termínmi. Vezmime si ako príklad dva konkrétne, napríklad „Naplánovať projekt“ a „Odladiť projekt.“ Na prvý pohľad vyzerajú byť v poriadku, oba majú však jednu zásadnú chybu – *akú*?

Pri projektovaní sa veľakrát dbá oveľa viac na stanovenie správnych termínov pre jednotlivé míľniky, ako sa dbá na ich definovanie, resp. pomenovanie. Človek pracujúci na projekte často nepresne odhadne stav splnenia míľnika pokiaľ nie je jasne definovaný. Napríklad míľnik „Naplánovať projekt“ je splnený po vytvorení počiatočného plánu a míľnik „Odladiť projekt“ je splnený prakticky neustále. To ale nie je pravda – plánovanie projektu prebieha neustále, nakoľko priebeh každého (nielen) softvérového projektu je značne dynamický, a preto je nutné neustále kontrolovať plnenie plánov a adekvátne tomu ich aj doplniť, prerobiť či zredukovať. Podobne, projekt je odladený až

keď „program úspešne prešiel všetkými sadami testovacích príkladov“ – to je korektné pomenovanie druhého míľnika.

Poučenie teda znie: treba dbať na to, aby každý míľnik mal okrem dobre odhadnutých termínov aj presne definovanú svoju podstatu, aby bolo jeho splnenie jednoducho a jednoznačne merateľné [4].

Žiadne „flákanie“

Avšak míľniky nie sú jediné, na čo treba pri tvorbe plánov dávať pozor. Správny projektový manažér neustále dbá aj na to, aby každý jeden člen tímu vedel čo má robiť – a to doslova [9]. Aj keď to znie príliš očividne, je nutné spomenúť aj túto radu.

Zadelenie naplánovaných úloh je obzvlášť dôležité hlavne pri rozsiahlych softvérových projektoch, pretože náročnosť rastie úmerne s veľkosťou projektu a projektového tímu. Neexistuje horšie premrhaný čas (i zdroje), než keď ľubovoľný člen tímu nemá čo robiť, prípadne keď viacerí tímoví kolegovia pracujú zbytočne a bezdôvodne na tej istej úlohe (existujú však výnimky, kedy to je potrebné – napríklad ak je daná úloha príliš náročná pre jedného; vtedy je ale otázne, či sa nedá táto úloha „rozbiť“ na menšie podúlohy zvládnuteľné samostatne). Azda najhorší možný prípad zlyhania zadelenia úloh je keď existuje úloha, na ktorej nepracuje nikto.

Na čo treba dbať?

Dosť bolo všeobecných rád, poďme si teraz zhrnúť konkrétne rady, ktoré sú užitočné pri plánovaní a súvisiacich činnostiach. Nižšie je uvedených niekoľko dobrých rád založených na radách Spolskeho [10], ktoré považujem za dôležité a výstižné:

- použite podporný nástroj na plánovanie primeraný veľkosti projektu,
- snažte sa vytvoriť čo najmenej komplikované plány (viď Obr. 2),
- rozdeľte každú úlohu na viaceré (čo najjednoduchšie) podúlohy,
- nechajte odhadnúť trvanie úlohy vývojára, ktorý sa jej venuje,
- priebežne sledujte splnenie úloh a aktualizujte naplánované doby ich trvania,

Schedule.xls

	1	2	3	4	5	6	7
1	Feature	Task	Priority	Orig Est	Curr Est	Elapsed	Remain
2	Spell Checker	Add Menu Item	1	12	8	8	0
3	Spell Checker	Main Dialog	1	8	12	8	4
4	Spell Checker	Dictionary	2	4	4	4	0
5	Grammar Checker	Add Menu Item	1	16	16	0	16
6							
7							
8							

Sheet1 Sheet2 Sheet3

Obr. 2. Príklad jednoduchého, no efektívneho plánu [10].

- berte do úvahy aj čas potrebný na integráciu a odladenie komponentov,
- neskracujte naplánované časy jednotlivých úloh za účelom vyvinutia časovej tiesne na projektový tím,
- ak projekt mešká, podľa možností buď vynechajte menej dôležité funkcie alebo predĺžte dobu potrebnú na dokončenie projektu.

Čomu sa vyhnúť?

Na druhej strane sú veci, ktorým sa treba pri tvorbe plánov vyslovene vyhýbať, pretože majú na plánovanie negatívny a kontraproduktívny vplyv. Zosumarizoval ich McConnell [8] a nižšie uvádzam tie, ktoré považujem za dôležité spomenúť:

- neplánovať alebo plánovať nedostatočne,
- nebrať do úvahy možné riziká projektu,
- používať ten istý plán na viaceré projekty,
- neprispôsobovať plán aktuálnemu stavu projektu,
- priskoro plánovať príliš podrobne,
- predpokladať, že časový sklz sa neskôr rýchlo dobehne,
- nepoučiť sa z predošlých chýb pri plánovaní.

Nedodržal sa plán!

V prípade, že náš projekt voči plánu mešká (v opačnom prípade, teda ak sa plán predbehol, netreba samozrejme robiť nič, prípadne možno oslavovať), netreba panikáriť. V panike totiž možno aj s tým najlepším úmyslom spraviť neefektívne, ba až deštruktívne rozhodnutia, ktorým treba, napriek veľkému nutkaniu, odolať.

Žiadni noví ľudia!

Ako jedno z najčastejších takýchto rozhodnutí, dokonca aj u skúsených manažérov, je rozhodnutie o pridelení nových pracovníkov do tímu riešiaceho meškajúci softvérový projekt so snahou dokončiť daný projekt načas [11]. Spôsobí to, práve naopak, ešte väčšie meškanie, pretože noví členovia tímu nie sú dostatočne oboznámení s projektom, ich tímoví kolegovia im musia pomôcť, projekt sa ešte viac omeškáva a úsilie vynaložené na pomoc novým tímovým členom prevýši celkové množstvo práce, ktoré do projektu priniesli. Pred touto situáciou už dávnejšie varoval Brooks, autor známeho Brooksovho zákona[1].

[1] Voľne preložený ako: „Pridanie ľudských zdrojov do meškajúceho projektu ho omeškáva ešte viac.“ [4]

Radšej to šéfovi povedzme...

Pracovne mladí manažéri projektov si v situácií, kedy im projekt, ktorý vedú, začína meškať, povedia, že to nevadí, ich tím to predsa dobehne a nepovažujú problém za dostatočne dôležitý, aby ho hlásili svojim nadriadeným. Načo by ich s tým zaťažovali? Veď ich nadriadení majú dosť iných starostí a koniec koncov – oni sú tam predsa na to, aby sa s tým vysporiadali. Na druhej strane sa takíto manažéri projektov boja, že ich nadriadení by začali sami tieto situácie riešiť, čo by, okrem iného, ublížilo ich autorite, prerušilo ich úsilie, a podobne. Takže nadriadení sa o meškaní jednotlivých projektov nedozvedia nič a všetko sa takpovediac „zametie pod koberec“ [4].

Vina v takýchto situáciách nie je, prekvapivo, väčšinou na strane projektových manažérov, ale práve na strane ich nadriadených, teda šéfov. Tí si zväčša sami môžu za to, že sa im ich podriadení „boja“ povedať zlé správy. Ako extrémny a odradzujúci prípad môže poslúžiť šéf, ktorého spomína Brooks [4]. Daný manažér zdvihol telefón a udelil rozkazy zamestnancom ihneď po prečítaní čo len prvej vety v správe o stave projektu...

Preto by si mali vrcholoví manažéri vytvoriť priateľskú atmosféru okolo seba a explicitne dať najavo svojím podriadeným projektovým manažérom, že sa im nebudú „miešať do roboty,“ pokiaľ si s ňou vedia poradiť sami. Takisto by mali šéfovia sami seba kontrolovať, aby nikdy neriešili problémy v prípade, že si iba overujú stav jednotlivých projektov.

Keď nič nejde podľa plánu...

Na záver si teda položme otázku: sú plány na to, aby sa porušovali? Dovolím si odpovedať nepriamo, a to príkladom [1], ktorý akoby bol priam stvorený pre tento účel.

Rozsiahly a prepracovaný plán pre vylodenie Spojencov s krycím označením Omaha Beach na severnom pobreží Francúzska dňa 6. júna 1944 vôbec nepredpokladal a ani nekontroloval priebeh samotného vylodenia. V skutočnosti takmer nič neprebiehalo podľa plánu. Nakoniec sa ale Nemcom toto územie ubrániť nepodarilo a zámer Spojencov ho obsadiť (a neskôr vyhrať samotnú 2. svetovú vojnu) vyšiel.

Ak posudzujeme misiu Omaha Beach podľa plánu, tak úplne zlyhala. Ale ak ju posudzujeme podľa cieľov, tak ide o jednoznačný úspech.

Softvérové projekty síce nie sú také nepredvídateľné ako vojenské operácie, ale predsa majú s nimi niečo spoločné. Je to skutočnosť, že aj plány v softvérovom inžinierstve do istej miery „neprežijú“ kontakt so samotným projektom i realitou. To však vôbec nevadí, pretože hlavným cieľom plánov je poskytnúť také smerovanie a rozdelenie dostupných zdrojov projektu, ktoré predstavuje jednu z možných ciest vedúcich k dosiahnutiu vytýčených cieľov.

Plány *boli* porušované, *sú* porušované a *budú* porušované. Nesmieme však zabúdať, že úspech projektov sa nemeria podľa (bezhlavého) dodržiavania plánov, ale podľa dosiahnutia cieľov projektu v rámci podmienok a ohraničení, ktoré boli naň stanovené.

Použitá literatúra

1. Armour, G.P.: The business of software: To plan, two plans. *Communications of the ACM*, Vol. 48, No. 9 (2005) 15-19.
2. Berntsson-Svensson, R., Aurum, A.: Successful software Project and Products: An Empirical Investigation. In *Proc. of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ACM Press, New York (2006), 144-153.
3. Brooks, F.P., Jr.: *No Silver Bullet: Essence and Accidents of Software Engineering*. IEEE Computer Society Press, Los Alamitos, 1987.
4. Brooks, F.P., Jr.: *The Mythical Man-Month: Essays on Software Engineering (Anniversary Ed.)*. Addison-Wesley Longman Inc., Boston, 1995.
5. Charette, R.N.: Why Software Fails. *IEEE Spectrum*, Vol. 42, No. 9 (2005) 42-49.
6. Churchville, D.: *Small Teams Make Better Software*. http://www.extremeplanner.com/blog/2006/08/small-teams-make-better-software.html [cit. 2007-10-25]
7. Kesteloot, L.: *Why Software is Late*. http://www.teamten.com/lawrence/writings/late_software.html [cit. 2007-10-25]
8. McConnel, S.: The Nine Deadly Sins of Project Planning. *IEEE Software*, Vol. 18, No. 5 (2001) 5-7.
9. Rettig, M., Simons, G.: A Project Planning and Development Process for Small Teams. *Communications of the ACM*, Vol. 36, No. 10 (1993) 45-55.
10. Spolsky, J.: *Painless Software Schedules*. http://www.joelonsoftware.com/articles/fog0000000245.html [cit. 2007-10-25]
11. Stellman, A.: *Late projects, man-months and the software crisis*. http://www.stellman-greene.com/2007/05/15/late-projects-man-months-and-the-software-crisis [cit. 2007-10-25]

Annotation

Are plans meant to be broken?

Software projects often end up being unsuccessful, fail or are cancelled. Projects of all dimensions are affected – ranging from small assignments created by students to vast international projects. The so-called “software crisis” still endures, since no one has yet invented an all-purpose solution that would once and for all put an end to software failures. There are multiple causes to that. This essay studies what roles planning and plan management plays in unsuccessful software projects of all sizes, clarifies the importance of accurate project planning, gives reasons why even the most perfect plans fail, and also gives practical tips on how to create the most robust “bulletproof” plan for (not only) a software project.