# Manažment v softvérovom inžinierstve

Zbierka esejí

Michal Tvarožek (editor)

# MANAŽMENT V SOFTVÉROVOM INŽINIERSTVE

# ZBIERKA ESEJÍ

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
2006

**Manažment v softvérovom inžinierstve**
**Zbierka esejí**

Editor a grafická úprava
Michal Tvarožek



| | |
|---|---|
| Martin Adam | Imrich Balko |
| Michal Barla | Peter Bartalos |
| Rudolf Dačo | Michal Habala |
| Ondrej Hluchý | Tomáš Klempa |
| Jan Porubský | Peter Sivák |
| Kristián Szobi | Michal Tvarožek |

Publikácia neprešla jazykovou úpravou.
Príspevky boli vytlačené v podobe dodanej autormi bez závažnejších zmien.

Koordinácia: Michal Barla
Cover design: Tomáš Klempa, Ján Porubský
Fyzická tvorba zborníka: Rudolf Dačo, Michal Habala
Elektronická prezentácia esejí: Martin Adam, Peter Sivák
Správa o rozdelení úloh: Peter Bartalos, Ondrej Hluchý
Kontrola kvality: Imrich Balko, Kristián Szobi

## Predslov

Tento zborník predstavuje zbierku esejí, ktoré vypracovali študenti v rámci predmetu Manažment v softvérovom inžinierstve na FIIT STU v akademickom roku 2005/2006. V jednotlivých esejach sa autori zamerali na problematiku manažmentu v softvérovom inžinierstve a na viaceré aktuálne problémy vyskytujúce sa pri riešení softvérových projektov malého až stredného rozsahu. Každá z esejí nielen uvádza čitateľa do konkrétnej problematiky ale aj na viacerých príkladoch prezentuje vlastné názory autora na konkrétne problémy. Aj keď si žiadna z esejí nekladie za cieľ vyčerpávajúcim spôsobom analyzovať jednotlivé aspekty softvérového inžinierstva, v každej z nich autori ponúkajú nový pohľad na existujúce skutočnosti a tým „svojou kvapkou“ prispievajú k riešeniam jednotlivých problémov. Práve v oboznámení čitateľa s problematikou a poskytnutí nového a „nepoškvrneného“ pohľadu na manažment v softvérovom inžinierstve vidíme hlavný prínos tejto zbierky esejí. Sme presvedčení, že mnohým pomôže pri rozšírení si znalostí o manažmente v softvérovom inžinierstve ako aj pri hľadaní ďalších zdrojov vedomostí z tejto oblasti.

Zbierku esejí sme rozdelili na tri bloky esejí, z ktorých každý nazerá na riešenú problematiku z mierne odlišného uhlu pohľadu. Prvý blok esejí zaujme najmä vyššie postavených manažérov z vedenia spoločnosti. Obsahuje eseje zaoberajúce sa problematikou tvorby softvéru z hľadiska firemnej kultúry a dlhodobejších cieľov spoločnosti. Venuje sa problematike produktivity tímov, kvality softvéru, riadenia vzťahov so zákazníkmi a organizácie softvérových tímov.

Druhý blok esejí je určený primárne projektovým manažérom, ktorých zaujme najmä aktuálnymi prístupmi k manažmentu rizík, odhadovaniu a sledovaniu postupu softvérových projektov. Nemenej zaujímavou je aj téma agilných metód, ktorých hlavné prínosy sa prejavujú práve pri riešení malých až stredných projektov. Úspešné využitie agilných metód však závisí nielen od schopností vývojárov ale aj od ich prijatia a presadzovania projektovými manažérmi.

Im je určený aj tretí blok esejí, ktorý rieši otázky tvorby a vývoja tímov, komunikácie v tíme a najmä otázky riešenia konfliktov v tíme. Práve problematika konfliktov a vývoja tímu je v súčasnosti mimoriadne dôležitá a eseje z tohto bloku majú čo povedať aj samotným členom softvérového tímu. Na vytvorenie úspešného tímu totiž nestačí „dať dokopy“ dobrých ľudí. Je potrebné aby títo ľudia spolupracovali na dosiahnutí spoločného cieľa – aby tvorili fungujúci a vysoko produktívny tím.

Veríme, že vás prehľad predloženej zbierky esejí zaujal a sme presvedčení, že každý v zbierke nájde aspoň jednu esej, ktorá ho zaujme a rozšíri jeho znalosti o problematike manažmentu v softvérovom inžinierstve.

Prajeme príjemné čítanie!

Michal Tvarožek
15. január 2006

# Obsah

# Úvod

V súčasnosti sa ľudská spoločnosť postupne mení zo spoločnosti priemyselnej na spoločnosť informačnú. Kľúčovými sa stávajú najmä informácie, ich spracovanie a vyhodnocovanie. Práve vo svetle týchto skutočností je dôležité si uvedomiť rolu informačných technológií a obzvlášť rolu softvéru ako takého.

Počiatky softvéru siahajú do druhej polovice dvadsiateho storočia, pričom samotný koncept softvéru navrhol Alan Turing v jednej zo svojich esejí. V časoch prvých počítačov softvér nezohrával takú veľkú úlohu ako v súčasnosti, pretože väčšina nákladov na počítačové systémy bola obsiahnutá v cene potrebného hardvérového vybavenia. Samotné softvérové systémy vtedy boli relatívne jednoduché a najmä značne obmedzené výpočtovou kapacitou hardvéru.

S narastajúcim časom a najmä s príchodom osobných počítačov a zvyšovaním ich výkonu sa začal pomer nákladov na softvér a hardvér obracať. V súčasnosti už náklady na softvér v mnohých prípadoch zásadne prekračujú cenu potrebného hardvéru, čo priamo súvisí s nárastom významu a zložitosti samotného softvéru, bez ktorého si súčasný svet už asi ani nevieme predstaviť.

Dôležitosť softvéru a najmä jeho špecifická povaha značne komplikujú jeho tvorbu. Je nevyhnutné si uvedomiť, že softvér je síce potrebné „vyrábať“, avšak samotná tvorba softvéru nadobúda podobu kreatívnej činnosti, čo spôsobuje že výsledky sú niekedy len veľmi ťažko merateľné. Problém ako merať kvalitu softvéru je možno podobný s problémom merania kvality obrazov. Vieme však jednoducho povedať, či kvalitnejšie obrazy maľoval Leonardo da Vinci alebo Vincent van Gogh?

Výrobný aspekt tvorby softvéru vstupuje do popredia v momente, keď si uvedomíme, že v súčasnosti je potrebná tvorba veľkého množstva softvéru pre najrôznejších zákazníkov, z ktorých každý má svoje špecifické požiadavky. Softvér je potrebné „vyrábať“ na mieru zákazníkom, pričom nemožno postupovať spôsobom, vlastným umelcom. Nie je reálne možné vytvoriť ľubovoľný softvér a dúfať, že sa niekomu zapáči a kúpi ho. Rovnako používatelia softvéru nemôžu čakať, kým niekto náhodou vytvorí vhodný softvér a umožní im si ho zakúpiť.

Softvér podlieha pravidlám trhu, ktorý núti softvérové spoločnosti k jeho cielenej tvorbe pre konkrétnych zákazníkov a k neustálemu zvyšovaniu kvality dodávaného softvéru ako aj k zvyšovaniu produktivity ich softvérových tímov. Uvedené skutočnosti si vynútili vznik disciplíny s názvom Softvérové inžinierstvo, ktorá sa zaoberá tvorbou a údržbou softvérových systémov pomocou systematického využitia technológií a skúseností z oblasti počítačovej vedy (angl. Computer science), projektového manažmentu, inžinierstva a ďalších.

Významným procesom v rámci softvérového inžinierstva je samotná tvorba softvérového systému zahŕňajúca celý jeho životný cyklus. Cieľom úspešnej softvérovej spoločnosti by malo byť nielen vytvoriť dobrý produkt, ale aj zabezpečiť aby každý nasledujúci bol ešte lepší. Týmto sa do popredia dostávajú procesy riadenia samotného softvérového procesu, ktorými sa zaoberá manažment v softvérovom inžinierstve a ktoré umožňujú neustále zlepšovať jeho kvalitu.

Predstavme si teraz založenie softvérovej spoločnosti, ktorá sa úspešne presadí na trhu, a ktorej vedúci manažéri budú mať dostatočné znalosti z odboru softvérového inžinierstva. Kľúčovým kapitálom tejto spoločnosti v budúcnosti nepochybne budú jej zamestnanci – softvéroví inžinieri, odberatelia – dlhodobí a spokojní zákazníci a dobré meno spoločnosti dané vysokou kvalitou dodávaných softvérových riešení.

Na dosiahnutie týchto zámerov si vedenie spoločnosti definovalo tri ciele: *kvalita*, *produktivita* a *zákazník*. Pri hľadaní spôsobov ako naplniť uvedené ciele sa manažéri inšpirovali esejami z prvého bloku esejí:

- Zlepšovanie produktivity softvérových tímov
- Manažment kvality a vplyv na výsledok projektu
- Manažment vzťahov so zákazníkmi
- Manažment softvérového projektu a organizácia softvérových tímov

Na základe poznatkov získaných z esejí *Zlepšovanie produktivity softvérových tímov* a *Manažment softvérového projektu a organizácia softvérových tímov* využili cielené získavanie nových zamestnancov s podporou ich profesionálneho rastu. Do firemnej kultúry zaviedli množstvo rozličných pozitívnych motivačných faktorov, čím naplno využili pracovný potenciál jednotlivých zamestnancov a vytvorili dobrú pracovnú atmosféru v rámci spoločnosti.

Táto skutočnosť sa pozitívne prejavila aj na kvalite dodávaných riešení, ktorej sa venuje esej *Manažment kvality a vplyv na výsledok projektu* a to tak z technického pohľadu procesu tvorby softvéru ako aj z pohľadu vhodnosti dodaného produktu pre uspokojenie potrieb zákazníka. Vďaka eseji *Manažment vzťahov so zákazníkmi* si spoločnosť včas vybudovala bázu znalostí o svojich zákazníkoch, ich potrebách, ziskovosti a rizikovosti, čo jej umožnilo úspešne sa orientovať na ich potreby, podporu a udržanie.

Vedenie spoločnosti sa však rozhodlo ísť ešte o krok ďalej a sprostredkovalo druhý blok esejí z tejto zbierky svojim projektovým manažérom. Viacerí z nich aj napriek prvotnému nedostatku nadšenia siahli po esejach:

- Manažment rizík v softvérovom projekte
- Odhadovanie v softvérových projektoch
- Sledovanie postupu softvérových projektov
- Agilné metódy vývoja softvéru a rozsah projektu

Veľmi ich zaujali aktuálne prístupy k identifikácii a riadeniu rizík ako aj spôsoby odhadovania parametrov softvérových projektov opísané v esejach *Manažment rizík v softvérovom projekte* a *Odhadovanie v softvérových projektoch*. Práve vďaka nim boli schopní pracovať efektívnejšie, vyvarovať sa najčastejším problémom a nájsť riešenia svojich každodenných problémov s riadením softvérových projektov. Dôsledné ukladanie a vyhodnocovanie údajov o predchádzajúcich projektoch im navyše umožnilo pracovať s omnoho väčšou istotou v úspešnosť projektov, ktorá viedla tak k spokojnosti zákazníkov, vedenia spoločnosti ako aj členov ich tímov.

Mnohých z nich zaujala tiež esej *Agilné metódy vývoja softvéru a rozsah projektu*, pretože im poskytla nový pohľad na spôsoby tvorby softvéru. Možnosť výberu

vhodného spôsobu vývoja konkrétnych softvérových systémov im umožnila znížiť neistotu a riziká spojené s projektmi. Výsledný softvér bol potom často kvalitnejší a prispôsobený na mieru zákazníkom.

Po úspechu, ktorý spôsobilo rozšírenie esejí z druhého bloku zbierky, vedenie spoločnosti podľahlo žiadostiam projektových manažérov a umožnilo im nahliadnuť aj do tretieho bloku esejí:

- Vývoj tímu v softvérovom projekte a vplyv na manažment
- Komunikácia v tíme
- Manažment konfliktov v tíme

Práve využitie princípov opísaných v esejach *Vývoj tímu v softvérovom projekte a vplyv na manažment* a *Manažment konfliktov v tíme* spôsobilo malú revolúciu v pracovnej atmosfére v spoločnosti. Pochopenie životného cyklu softvérových tímov a príčin vzniku konfliktov v nich umožnilo projektovým manažérom minimalizovať množstvo problémov spôsobených nedorozumeniami a problémami s komunikáciou. Čas, ktorý bol v minulosti potrebný na ich riešenie bolo zrazu možné využiť na ďalšie zvýšenie produktivity tímov, resp. kvality softvéru ako aj na zaslúžený odpočinok.

S narastajúcou veľkosťou spoločnosti začali prácu softvérových tímov čoraz častejšie sťažovať problémy spojené s komunikáciou ich členov. Aj keď sa mnohí z nich nikdy osobne nestretli, pretože pracovali na rozličných pracoviskách, potrebovali spolupracovať na dosiahnutí spoločných cieľov. Vďaka poznatkom nadobudnutým z eseje *Komunikácia v tíme* sa však vždy podarilo projektovým manažérom nájsť rýchly a efektívny spôsob komunikácie jednotlivých členov tímu.

Prezieravosť zakladateľa spoločnosti, ktorý naplno využil predložený zborník esejí a pretavil myšlienky jednotlivých esejí do praxe sa prejavila v úspechu spoločnosti na trhu a nadšení zamestnancov pre tvorbu kvalitného softvéru pre spokojných zákazníkov. Keď sa na oslavách dvadsiateho výročia založenia spoločnosti jeden novinár spýtal jej zakladateľa na tajomstvo jeho úspechu, ten opatrne vybral zborník esejí zo svojho kufríka a so širokým úsmevom na tvári odpovedal: „V tomto zborníku nájdete všetko, čo potrebujete vedieť. Ostáva vám len doplniť zdravý podnikateľský duch a nadšenie pre tvorbu softvéru.“

Predstava opísanej softvérovej spoločnosti je v súčasnosti iste idealistická. Rovnako predložený zborník esejí podrobne nevysvetľuje všetky aspekty manažmentu v softvérovom inžinierstve. Pokladáme ho však sa významný príspevok k tomu, aby sa softvérové spoločnosti v budúcnosti čo najviac priblížili opísanému ideálu.

# Manažment v softvérovom inžinierstve z hľadiska vedenia spoločnosti

# Zlepšovanie produktivity softvérových tímov

PETER SIVÁK

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
sivak01@student.fiit.stuba.sk

**Abstrakt.** Produktivita je definovaná ako pomer veľkosti výstupu v podobe tovarov a služieb a množstva vstupov potrebných na vytvorenie tohto výstupu. Je veľmi dôležité, aby manažéri v softvérových projektoch dobre poznali metódy, ktoré vedú k významnému zlepšeniu produktivity. V eseji opisujem niektoré z týchto metód použitím systematického prístupu k ich kategorizácii.

## Úvod

Vývoj softvérových systémov je podobne, ako veľká časť iného ľudského snaženia, výrazne ovplyvnený ekonomickými parametrami. V oblasti vývoja týchto systémov môžu byť úspešné iba spoločnosti, ktoré dokážu minimalizovať čas potrebný na pretransformovanie úvodnej myšlienky do softvérového systému a dodať výsledok v čo najkratšom čase. Osobitne dôležitou je produktivita v jasne definovaných projektoch, v ktorých sa spoločnosti medzi sebou nedokážu výraznejšie odlíšiť iným parametrom. Snahou manažmentov softvérových spoločností sa preto nevyhnutne stáva *kontinuálne zvyšovanie produktivity* svojich tímov.

Produktivita je priamo prepojená so životným štandardom softvérových inžinierov. Ak sa zvyšuje zisk vytvorený za jednu hodinu práce, zvyšujú sa aj príjmy. Ak produktivita klesá, príjmy sa tiež znižujú. V prípade, že sa konkurujúce spoločnosti vo všetkých iných aspektoch vyrovnajú, tak produktívnejšia spoločnosť bude mať vyšší zisk. Kľúčom k pretrvávajúcemu rastu miezd v softvérovom priemysle je preto aj každoročné zvyšovanie produktivity.

Cieľom tejto eseje je zhrnúť viacero metód na systematické zvyšovanie produktivity pri vývoji softvérových systémov. Produktivita sa všeobecne definuje ako *množstvo vytvoreného výstupu za jednotku času*. V komerčnom prostredí spoločností, ktoré vyvíjajú a predávajú softvér ako produkt, sa produktivita často definuje ako vygenerovaný zisk na jednu hodinu práce zamestnanca alebo ako vygenerovaný zisk na každú investovanú korunu. Spoločnosti, ktoré vyvíjajú softvér pre svoju potrebu, môžu pod produktivitou rozumieť priemerné zvýšenie zisku spoločnosti dosiahnuté z jednej pracovnej hodiny vývojára.

*Manažment v softvérovom inžinierstve, december 2005, s. 7-12.*

K zlepšovaniu produktivity môžeme pristupovať v zásade dvomi spôsobmi. Jeden je založený na jej priamom meraní. Meranie produktivity nám umožňuje hocikedy objektívne posúdiť, či sa aplikované zmeny prejavili na výslednej produktivite pozitívne alebo negatívne, a následne podľa týchto zistení robiť ďalšie úpravy. Druhý spôsob počíta s používaním existujúcich najlepších postupov v danej oblasti a nepožaduje dôsledné vyhodnocovanie produktivity. Tento spôsob je užitočný najmä v malých softvérových spoločnostiach. Väčšie spoločnosti by mali svoju produktivitu vedieť vyčísliť.

## Meranie produktivity

Ak chceme zvyšovať produktivitu softvérových tímov alebo ich členov je dobré, ak poznáme jej aktuálny stav a trend vývoja. Spoľahlivé meranie produktivity je veľmi náročné a často až nemožné. Dlho sa v softvérom priemysle používalo meranie produktivity podľa počtu riadkov vo výslednom kóde (LOC; lines of code). Viedlo to samozrejme k umelému naťahovaniu programov o riadky, ktoré sú zbytočné. Táto metóda neumožňuje rozlíšiť neproduktívne činnosti, ako napríklad písanie testovacích zdrojových kódov. Známi je tiež výrok Edsgera Dijkstru, ktorý hovorí, že ak chceme počítať riadky v kóde, nemali by sme ich považovať za riadky vyprodukované, ale za riadky minuté.

Pri meraní produktivity si tiež treba uvedomiť, že súčasťou softvérových tímov nie sú iba programátori. Je potrebné merať aj produktivitu ďalších členov tímu – analytikov, návrhárov a pod. Na úspešné meranie produktivity potrebujeme metriku, ktorá nepodporuje vývojárov v produkovaní množstva kódu bez skutočnej hodnoty. Jednou možnosťou je merať počet implementovaných funkčných celkov, prípadne merať priamo zisk vygenerovaný za hodinu práce vývojára. Použitie posledne menovanej metódy je možné iba v špecifických prípadoch – napríklad pri údržbe systémov.

Myslím si, že dobrým spôsobom je merať produktivitu programátorov podľa počtu vyriešených úloh. Musí však byť splnená podmienka, že navrhnuté úlohy sú dostatočne malé, aby sa každá úloha dala ohodnotiť niekoľkostupňovou stupnicou – napríklad: málo náročná, stredne náročná a veľmi náročná úloha. Ku každému stupňu sa určia váhy. Sčítaním váh všetkých vykonaných úloh za nejaký čas sa získa číselné vyjadrenie produktivity vývojára. Rozbitie veľkých úloh do menších okrem toho pomôže aj pri plánovaní softvérového projektu.

## Zlepšovanie produktivity

Predtým, ako preskúmame rôzne metódy na zlepšovanie produktivity samotných softvérových tímov, pozrime sa na to, ako vplýva štruktúra softvérových projektov v spoločnosti na jej celkovú produktivitu, v zmysle veľkosti vytvoreného zisku v pomere ku vstupným nákladom. Projekty sa v komerčnej sfére spravidla rozdeľujú na také, ktorých cieľom je vytvorenie nového systému, a na také, ktoré udržiavajú

existujúce systémy. Podľa [5] je rozumnejšie orientovať spoločnosť viac na vyvíjanie nových systémov, ako na udržiavanie existujúcich. Spravidla vyšší zisk pri vývoji nových produktov je dôsledkom toho, že sa v nich často zhmotňujú nové a inovatívne riešenia, ktorých cena sa stanovuje prvýkrát. Na druhej strane, pôvodné systémy sú postavené na technológiách, ktoré sú náročnejšie na údržbu.

Produktivitu zlepší zníženie množstva vstupov a/alebo zvýšenie množstva výstupov. Množstvo vstupov môžeme zmenšiť znížením miezd vývojárov alebo vykonávaním menšieho množstva práce. Najschodnejšie je vykonávanie menšieho množstva práce tým, že budeme vytvárať iba to, čo zákazník skutočne potrebuje. Zo štatistík [5] totiž vyplýva, že osemdesiat percent hodnoty väčšiny systémov je obsiahnutých v dvadsiatich percentách funkcií a až dve tretiny funkcií systémov je používaných veľmi málo, prípadne vôbec. Aby sme vytvárali naozaj iba to, čo zákazník potrebuje, môžeme ho napríklad motivovať v tom, aby formuloval čo najviac požiadaviek na nový systém už v prvých fázach projektu.

Prejdime teraz k metódam na zvyšovanie produktivity softvérových tímov. Podľa R. Chianga a V. Mookerjeeho [2] si zvyšovanie produktivity vývoja softvéru vyžaduje vyvážený prístup k trom základným pilierom manažmentu v softvérovom inžinierstve:

- k technológiám,
- k ľuďom a
- k procesu vývoja.

V deväťdesiatych rokoch sa produktivita zvyšovala najmä zlepšovaním technológií používaných pri vývoji softvérových systémov. Rovnako sa v tomto období darilo zvyšovať produktivitu vďaka veľkému záujmu zákazníkov nasadzovať nové technológie. Táto dôvera zákazníkov v nové technológie sa neskôr začala prudko znižovať, čo spôsobilo znižovanie ziskov a následné zatváranie menej výkonných softvérových spoločností.

Ako sa softvérový priemysel vyvíja, je čoraz ťažšie zvyšovať produktivitu zlepšovaním technológií. Napriek tomu sa v tejto súvislosti ako zaujímavá oblasť výskumu javí automatické generovanie zdrojového kódu pre dobre definované čiastkové problémy. Príkladom sú generátory objektovo-relačných mapovačov, ktoré odbremeňujú programátora od písania kódu, ktorý transformuje dáta z relačnej databázy do ich objektovej reprezentácie a naopak.

## Ľudia a ich pracovný potenciál

Úspech softvérovej spoločnosti závisí najmä na kvalifikovanosti jej zamestnancov. Každý člen softvérového tímu musí mať dostatočné vedomosti a zručnosti na to, aby dokázal efektívne využívať dostupné technológie a vykonávať definované procesy. Úlohou riadiacich pracovníkov je vytvoriť priestor na zlepšovanie vedomostí, zručností, motivácie a výkonnosti členov softvérových tímov. Zvládnuť manažment a kontinuálny rozvoj pracovnej sily je náročné. Preto vznikol súbor odporúčaní P-CMM (People Capability Maturity Model) [3], ktorý detailne opisuje postup, akým by sa mali

realizovať organizačné zmeny. Tie sa musia plánovať dlhodobo, lebo sa na produktivite prejavujú až v dlhšom časovom horizonte.

Všetky metódy na zvyšovanie kvalifikácie pracovníkov pracujú s množinou zamestnancov, ktorí sú v spoločnosti. Na produktivitu preto výrazne vplývajú spôsoby, akým spoločnosť hľadá nových zamestnancov, akým si vyberá medzi uchádzačmi o zamestnanie, ako ich obsadzuje do pracovných pozícií a ako ich presúva na nové pozície. Úspešné prístupy napríklad zahŕňajú cielené získavanie nových pracovníkov z radov absolventov a ich profesionálne nasmerovanie na oblasť, v ktorej spoločnosť v budúcnosti predpokladá najväčší nedostatok.

Spôsob obsadzovania nových pracovných miest spravidla vychádza z kultúry danej spoločnosti a z jej dlhodobej stratégie. Častým problémom pri získavaní nových pracovníkov je podľa mňa taká politika organizácie, ktorá nedovoľuje ohodnotiť nových pracovníkov rovnako ako dlhoročných s podobnými vedomosťami. Takýto prístup znižuje motiváciu nových pracovníkov dosahovať vysoké výkony, a teda znižuje potenciálnu produktivitu.

Ďalšou významnou oblasťou manažmentu ľudských zdrojov, ktorá má veľký vplyv na produktivitu pracovníkov, je oblasť *komunikácie a koordinácie*. Jej cieľom je zabezpečiť pravidelnú a efektívnu výmenu informácií v organizácii. Rovnako by mala zabezpečiť, aby každý zamestnanec mal vedomosti a zručnosti nutné na efektívne zdieľanie informácií a na efektívnu koordináciu aktivít s ostatnými členmi tímu. Jednotliví zamestnanci by mali vedieť rozpoznať a predchádzať takým druhom komunikácie, ktoré vedú k vytváraniu nepriateľského prostredia pre iných zamestnancov. Je to napríklad nepriznanie rovnakých šancí na sebarealizáciu, diskriminácia a podobne.

Produktivitu ovplyvňuje aj *fyzické prostredie* v ktorom sa zamestnanec nachádza. Ide napríklad o dostatočné osvetlenie pracoviska či vytvorenie tichého prostredia, ktoré umožňuje pracovníkom lepšie sa sústrediť na svoju prácu. Podľa mojich skúseností je udržiavanie tichého prostredia v softvérových firmách často zanedbávané najmä zo strany manažérov. Tí s cieľom detailne poznať problémy, ktoré riešia vývojári, strávia komunikáciou s nimi veľa času, neuvedomujúc si, že tým rušia aj ostatných nachádzajúcich sa v rovnakej kancelárii.

Na zvyšovanie produktivity je tiež nevyhnutné vybudovať v spoločnosti systém, ktorý umožní pravidelné školenie jednotlivých vývojárov v oblastiach, v ktorých riešia svoje úlohy.

## Motivácia pracovníkov

Motivácia je okrem iného aj prevenciou proti problémom s produktivitou a kvalitou práce [1]. Každého zamestnanca motivuje niečo iné. Preto neexistuje jednoduchý návod, ktorý by umožňoval motivovať všetkých zamestnancov. Hľadanie motivačných faktorov, ktoré motivujú jednotlivých zamestnancov, nie je jednoduché. Zamestnanci si často svoje motivačné faktory ani sami neuvedomujú. Keď si ich aj uvedomujú, neradi ich svojim nadriadeným prezrádzajú. Tí často predpokladajú, že ich zamestnancom ide iba o peniaze, a preto sa ani nepokúšajú hľadať iné motivačné faktory. Podľa [4]

k najčastejším motivačným faktorom popri mzde patrí uznanie za dobre vykonanú prácu, poskytnutie väčšej dôvery a samostatnosti, širšie vedomie významu práce a umožnenie pružnejšieho časového usporiadania pracovného dňa.

Motivačné metódy rozdeľujeme na pozitívne a na negatívne. Negatívna motivácia je zvyčajne nejaká forma trestu prichádzajúca na rad v prípade, ak zamestnanec niečo zanedbal alebo nevykonal. Pozitívna motivácia je nejaká forma odmeny za dobre vykonanú prácu. Z dlhodobého hľadiska možno dosiahnuť oveľa vyššiu výkonnosť zamestnancov, keď pred negatívnymi motivačnými metódami uprednostníme pozitívne.

Opakujúcim sa problémom pri pozitívnej motivácií je odmeňovanie rôznych pracovných výkonov rovnakou odmenou. Okrem toho, že je to zjavne nespravodlivé, to v konečnom dôsledku znižuje motiváciu minimálne toho pracovníka, ktorý podal vyšší výkon. Tiež sa tým v organizácii vytvára nepriateľské prostredie, ktoré sa neskôr môže odraziť na celkovej produktivite softvérového tímu.

## Zlepšovanie procesu vývoja

Tretí spôsob zvyšovania produktivity softvérového tímu spočíva v zlepšovaní samotného procesu vývoja softvérového systému. Bez využitia dobrého procesu, sa vývoj systému môže stať chaotickým, čo sa následne odrazí v celkovo nízkej produktivite vývoja a v nízkej kvalite systému.

Základnou otázkou, ktorej riešenie je súčasťou definovania procesu vývoja, je kedy sa má vykonávať koordinácia jednotlivých členov tímu. Existujú tri základné prístupy ku koordinácií:

- Big Bang,
- periodická integrácia a synchronizácia,
- integrácia riadená výskytom chýb.

Pre každý projekt musí manažér osobitne zvažovať, aký prístup ku koordinácií členov tímu a k integrácii komponentov použiť. Niekedy sa toto rozhodnutie musí zmeniť aj počas priebehu projektu, a to najmä v prípadoch, keď sa vyskytnú nečakané problémy.

Hlavnou myšlienkou prístupu *Big Bang* je ponechávanie koordinácie a integrácie na koniec projektu. Výsledný produkt sa vytvorí až po dokončení a otestovaní všetkých jeho súčastí. Tento prístup vychádza zo známeho vodopádového modelu. Vývojári nie sú počas trvania projektu zbytočne zdržovaní komunikáciou a môžu sa sústrediť na svoju prácu. Vyžaduje si to však od nich vysokú schopnosť samostatne riešiť vzniknuté problémy. Preto je tento prístup vhodný najmä v tímoch so skúsenými a vysoko motivovanými vývojármi, prípadne pri projektoch s dobre definovanými úlohami. Napriek tomu, že je počas projektu možné zmeniť prístup ku koordinácii za iný, často je náročné zistiť, kedy nastal ten správny okamih, keď už vieme, že prístup Big Bang prináša viac problémov ako úžitku.

Periodická integrácia a synchronizácia vychádza z iteratívneho modelu vývoja softvéru. Spočíva v definovaní intervalov, v ktorých sa jednotlivé časti systému integrujú. Dĺžka týchto intervalov závisí od druhu projektu, prípadne od toho, ako

často potrebujeme poznať aktuálny stav projektu a kvalitu výsledku. Je to najčastejšie používaný prístup ku koordinácii.

Pri takých typoch projektov, s ktorými už tím má určité skúsenosti, je vhodné využiť prístup s integráciou riadenou výskytom chýb. Pri takomto prístupe sa integračné stretnutia zvolávajú až pri výskyte problémov. Nevýhodou je trochu menší prehľad o stave projektu ako pri periodickej integrácii. Ten je však vyvážený znížením množstva neefektívnej komunikácie v tíme.

## Záver

Je dôležité zaoberať sa systematickým zvyšovaním produktivity pri vývoji softvérových systémov v tímoch. Dôvodom je ostrá konkurencia medzi softvérovými spoločnosťami, ktorá neumožňuje dlhšie pôsobenie na trhu tým spoločnostiam, ktoré produktivitu neberú vážne.

Tento príspevok sumarizuje dôvody, ktoré vedú organizácie k tomu, aby sledovali vývoj produktivity svojich softvérových tímov a opisuje metódy na priame či nepriame meranie produktivity. Príspevok vymenúva a podrobnejšie rozoberá oblasti, ktoré môže manažment ovplyvniť s cieľom zvýšiť produktivitu.

## Použitá literatúra

1. Bieliková, M.: *Softvérové inžinierstvo*. Bratislava, 2000.
2. Chiang, I. R., Mookerjee, V. S.: Improving Software Team Productivity. In *Communications of the ACM*, Vol. 47, No. 5 (May 2004), p. 89-93.
3. Curtis, B., Hefley, W., Miller, S.: *People Capability Maturity Model*. Carnegie Mellon, Pittsburgh, 2001.
4. Chyby pri odmeňovaní a motivácii zamestnancov. *Infoware*, October 2005, p. 19
5. Poppendieck, M.: The software productivity imperative. In *Agile Project Management*, Vol. 5, No. 3.

### Annotation

*Improving software team productivity*

Productivity is the output of goods and services divided by the inputs needed to generate that output. It is essential that managers in software projects have a fundamental understanding of the methods for significantly improving productivity. In this essay we describe some of these methods using a systematic approach to categorize them.

# Manažment kvality a vplyv na výsledok projektu

ONDREJ HLUCHÝ

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
hluchy99@student.fiit.stuba.sk

**Abstrakt.** Vývoj dobrého softvérového systému je veľmi komplexná úloha. Ak chceme vytvoriť dobrý softvérový produkt, musíme prihliadať na niekoľko metrík kvality softvéru. Zložitosť systému hrá významnú úlohu v procese kontroly a riadenia kvality, pretože vo všeobecnosti vplýva na metriky ako spoľahlivosť, testovateľnosť a udržovateľnosť softvéru. Preto zabezpečenie kvality softvéru treba vziať do úvahy počas celého životného cyklu vývoja softvéru s ohľadom na nové stratégie, nástroje, metodológie a techniky.

Táto esej je prehľadovou štúdiou, ktorej cieľom je poskytnúť ucelený pohľad na manažment kvality v procese vývoja softvéru. Autor v nej odpovedá na otázky, čo to vlastne je manažment kvality a prečo je dôležitý. Ďalej porovnáva existujúce modely a štandardy na zabezpečenie kvality. V závere opisuje vplyv kvality na výsledok projektu na základe vlastných skúseností z praxe.

## Úvod

Náklady na neúspešné IT projekty v USA boli nedávno odhadnuté na 84 miliárd dolárov len za jediný rok [1], takže na kvalite softvéru záleží v dnešnej dobe viac ako kedykoľvek predtým. A záleží na nej nám všetkým, pretože my používame ten softvér. Naša závislosť na softvéri sa zvyšuje z roka na rok či si to uvedomujeme alebo nie. Každým dňom viacej z nás používa softvér na viacero úloh v oblasti informačných technológií (IT), informačných systémov (IS), vo vnorených systémoch v spotrebných tovaroch ako sú napríklad mobilné telefóny a samozrejme v Internete. Riziká spojené so zlyhaním softvéru sa zvyšujú s používaním softvéru. To zahrňuje väčšie riziká pre organizácie, ak softvér zlyhá alebo je nevyhovujúci, a väčšie sklamanie alebo ujmu jednotlivcov, ak softvér nesplní ich očakávania.

Medzitým sa v posledných rokoch zvýšili tlaky na moderné organizácie vrátane firiem. Tieto tlaky – znižovanie nákladov, maximalizácia zisku, zvýšené očakávania zo strany zákazníkov, globálne komunikácie, neustále zmeny a potreba hľadať nové trhy –

*Manažment v softvérovom inžinierstve, december 2005, s. 13-19.*

sa stávajú tlakmi na softvérové tímy, aby produkovali viac softvérových produktov, rýchlejšie, so zvýšenými očakávaniami na funkcionalitu softvéru.

Prečo nás IT tak často sklamú? Prečo sa softvér nevytvára bez chýb? Jedna príčina je celkom jednoduchá: IT systémy sú vytvárané ľuďmi a ľudia robia chyby. Toto platí pre každú ľudskú aktivitu, ale v IT máme množstvo okolností, ktoré k chybám napomáhajú:

- IT systémy často vytvárajú tímy ľudí, ktorí nebudú vytváraný systém používať. Za týchto okolností slabá komunikácia medzi tímami a budúcimi používateľmi zvyšuje šance na chyby.
- IT oddelenia sú často neschopné prispôsobiť vytváraný softvér kultúre, procesom alebo cieľom v oblasti, pre ktorú je softvér určený.
- IT systémy sú zložité a ich zložitosť sa stále zväčšuje. Tiež interkomunikácia s inými systémami je stále zložitejšia.

Iba posledná okolnosť je technický problém. Ostatné vymenované okolnosti sa týkajú ľudí, ich schopnosti komunikovať a navzájom si porozumieť a schopnosti učiť sa od druhých a z vlastných skúseností.

## Definície kvality softvéru

Podľa orientácie môžeme vyčleniť štyri rozdielne definície kvality. Tieto sú *produktovo*-orientovaná, *vývojovo*-orientovaná, *používateľsky*-orientovaná a *hodnotovo*-orientovaná [1].

Dve definície kvality podporované IT ľuďmi sú produktovo-orientovaná a vývojovo-orientovaná definícia. V projektoch dávame prednosť definíciám, ktoré nám umožnia merať pokrok a úspech. Chceme fixovať kvalitu na niečo, čo je dodateľné a merateľné.

- V definícii orientovanej na produkt je kvalita založená na dobre definovanej sade metrík, ktoré treba merať objektívnym a kvantitatívnym spôsobom. Môžeme z nich odvodiť akceptačné kritériá, aby sme vedeli objektívne posúdiť kvalitu dodaného produktu. *Príklad: Štandardy ako napríklad ISO 9126 definujú atribúty spoľahlivosť, použiteľnosť, bezpečnosť a funkcionalita spolu s metrikami pre ne. "Softvér je spoľahlivý na 98% ak beží nepretržite viac ako 7 dní. Čas na zotavenie je menší ako 1 minúta po každom zlyhaní."*
- Definícia orientovaná na vývoj sa sústreďuje na vývoj softvérových produktov, to znamená ich špecifikáciu, návrh a implementáciu. Kvalita závisí od rozsahu implementovaných požiadaviek. Úspech sa meria ako schopnosť sledovať proces a dodať produkt s ohľadom na dohodnutú špecifikáciu. Produkt budeme verifikovať (je systém správny podľa špecifikácie?), ale ak nezoberieme do úvahy definíciu kvality orientovanú na používateľa (viď nižšie), môžeme zabudnúť validovať (je toto správna špecifikácia?). *Príklad: Opakovateľný, kontrolovateľný proces, ktorý vyhovuje špecifikácii. "Softvér bol vytvorený podľa špecifikácie a obsahuje malé množstvo chýb."*

Existujú dve ďalšie definície kvality, ktoré odrážajú pohľady používateľa a zákazníka. Pretože tlaky na organizácie sa časom menia, to čo predstavuje "kvalitu", sa môže tiež časom meniť. Niekedy sú zmeny taktické – *"Musíme znížiť náklady v tomto štvrťroku!"* – a niekedy môžu byť strategické – *"Chceme byť lídrami na trhu!"* – a tieto si môžu protirečiť.

- Používateľsky orientovaná definícia hovorí, že kvalita vyjadruje vhodnosť použitia. Kvalita softvéru by mala byť stanovená používateľom produktu v špecifickej situácii. Rozličné obchodné charakteristiky vyžadujú rozličné typy softvérových produktov; nielen na robenie iných vecí, ale tiež treba dbať na to, že rozliční ľudia môžu vykonávať svoje úlohy odlišne. Táto skutočnosť môže byť subjektívna a nedá sa určiť len na základe kvantitatívnych metrík. Práve používateľsky orientovaná definícia nás nabáda k validácii ako aj verifikácii systému. *Príklad: Vhodnosť pre cieľ. "Môžem robiť svoju prácu efektívne ak použijem tento softvér. "*
- Hodnotovo orientovaná definícia je zameraná na veci, ktoré majú vplyv na firmu ako celok. Kvalita softvéru by mala byť vždy určovaná prostredníctvom úsilia a cenových aspektov. To sa deje komunikáciou všetkých zúčastnených strán: sponzori, zákazníci, vývojári a výrobcovia. *Príklad: Návrat investícií. "Ak vydáme softvér teraz, budeme musieť vynaložiť $250000 navyše na podporu v prvom mesiaci. Ak sa oneskoríme o mesiac, bude to stáť organizáciu $1000000 na pokutách a stratu biznisu. Mali by sme softvér vydať alebo ďalej testovať?"*

V jednom projekte môžeme brať do úvahy niekoľko definícií kvality a nemusia si to ani uvedomiť všetci ľudia v projekte. Je dôležité uvedomiť si, že neexistuje žiadna správna definícia kvality. To ako zadefinujeme kvalitu pre určitý produkt, službu alebo projekt závisí od situácie.

## Fundamentálne koncepcie dokonalosti

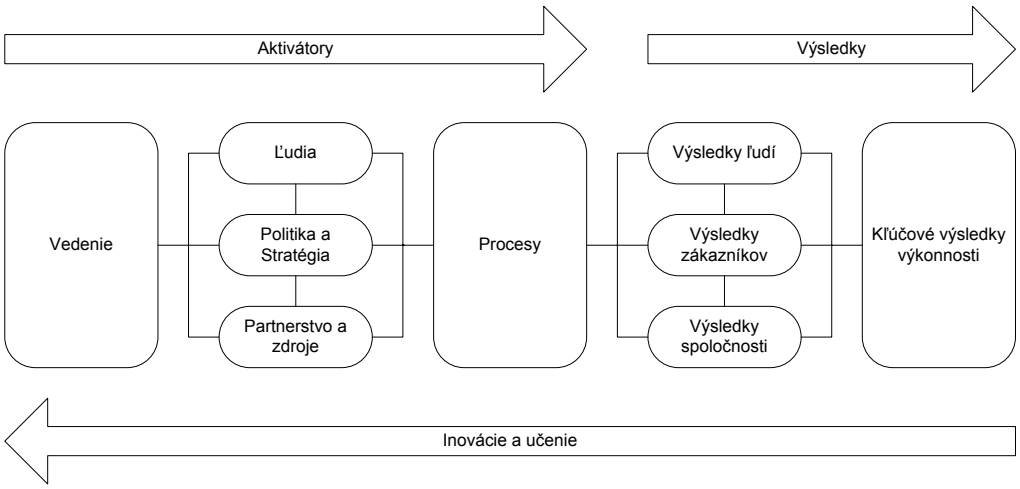
Tieto koncepcie [1] sa používajú v množstve modelov vrátane EFQM (European Foundation for Quality Management) Modelu dokonalosti, ktorý sa používa vo väčšine krajín Európskej Únie. Podobné organizačné modely založené na týchto koncepciách boli vyvinuté aj v iných krajinách.

Niektoré modely definujú kvalitu procesov pre jednotlivcov, iné pre tímy a niektoré sú určené pre organizácie. Najznámejšie z týchto modelov sú:

- V Európe známy rámec pre organizačnú dokonalosť EFQM Model Dokonalosti, ktorý je podobný iným, napr. Malcolm Baldrige Model v USA;
- Dva štandardy pre procesy – ISO 9000:1994 a ISO 9000:2000;
- Skupina modelov IT zrelosti združená okolo CMM (Capability Maturity Model) a dva modely pre implementáciu CMM: TSP (Team Software Process) a PSP (Personal Software Process).

## EFQM Model Dokonalosti

EFQM (European Foundation for Quality Management) Model Dokonalosti [1] je organizačný model, ktorý používa nenormatívny rámec. Nie je to výslovne IT rámec. Môže sa použiť v organizáciách akejkoľvek veľkosti a typu a je určený pre korporácie, spoločnosti alebo nezárobkové organizácie. EFQM poskytuje rámec pre dokonalosť v deviatich kritériách. Päť z nich sú tzv. „Aktivátory" pre dokonalosť a štyri sú metriky pre "Výsledky" (**Obr. 1**).



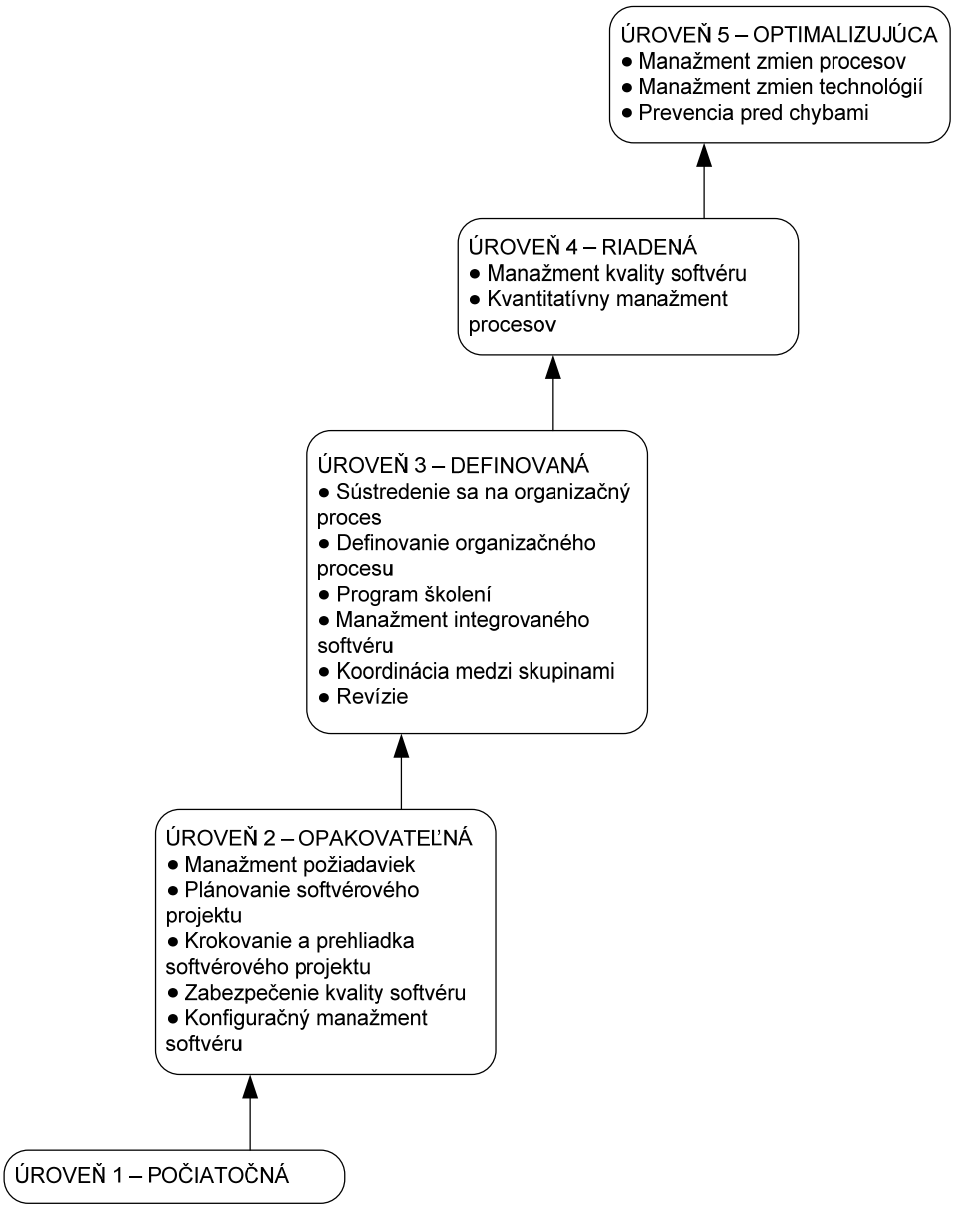

**Obr. 1.** EQFM Model Dokonalosti.

## ISO 9000:1994 and ISO 9000:2000

ISO 9000:1994 je štandard na zabezpečenie kvality v návrhových a výrobných procesoch. Tento štandard je prísny vo svojej definícii a interpretácii kvality. Definuje veľké množstvo procesov, ktoré treba dodržiavať s ohľadom na audity a evidenciu pričom nezáleží na tom, či tieto procesy poskytnú zákazníkovi alebo projektovému tímu ten najlepší výsledok. [2] V ISO 9000:2000 je menšie množstvo definovaných a dokumentovaných procesov a väčší dôraz na ľudí, aby porozumeli úlohám, ktoré musia vykonať a väčší dôraz na uspokojenie zákazníka. ISO 9000:2000 zahŕňa neustále zlepšovanie a približuje sa k EFQM. Hoci ISO 9000 štandardy spĺňajú niektoré Fundamentálne Koncepcie Dokonalosti, sú v nich významné medzery, špeciálne v ISO 9000:1994.

## Modely IT zrelosti – CMM a príbuzné modely

Koncept organizačnej zrelosti a spôsobilosti IT organizácií bol vyvinutý v SEI (Software Engineering Institute) a definuje lepší spôsob vývoja softvéru [1]. Softvér považuje za inžiniersku disciplínu a organizácie delí do piatich úrovní CMM (Capability Maturity Model) (**Obr. 2**) [3]:




**Obr. 2.** Úrovne zrelosti v CMM.

- Úroveň 1 – Počiatočná: Projekty sú chaotické a ad hoc. "Každý má svoj vlastný proces."
- Úroveň 2 – Opakovateľná: Požiadavky sú manažované a projekty sa realizujú podľa dokumentovaných plánov. "Každý tím má svoj vlastný proces; tímy môžu opakovať svoju prácu."

- Úroveň 3 – Definovaná: Softvérové inžinierstvo a riadiace procesy sú stabilné a ostanú také aj pod stresom. "Každý tím v organizácii používa ten istý proces; môžeme sa začať zaoberať zmenami."
- Úroveň 4 – Riadená: Organizácia riadi svoje procesy kvantitatívne a meria výkonnosť a kvalitu vo všetkých projektoch. "Celá organizácia sa meria, takže VIEME povedať ako sa nám darí."
- Úroveň 5 – Optimalizujúca: Neustále zlepšovanie a aktívne riešenie nedostatkov. "Môžeme stavať na našom poznaní a ďalej sa zlepšovať."

Pokrok cez tieto úrovne sa meria pomocou kľúčových indikátorov procesu. Všetky indikátory na jednej úrovni musia byť splnené predtým ako môžeme povedať, že organizácia je na tej úrovni. Úrovne a kľúčové indikátory sa sústreďujú na proces vývoja softvéru a meranie toho procesu.

CMM poskytuje odstupňovaný postup k zlepšeniu IT procesov. Základná premisa hovorí: IT organizácie potrebujú najprv kráčať predtým ako budú behať. Dômyselné inžinierske a meracie procesy nemôžu fungovať, pokiaľ nie sú postavené na systéme silných základných riadiacich praktík.

CMM pokrýva vývoj softvéru a testovanie považuje za časť vývoja, pričom na testovanie kladie explicitné požiadavky na úrovni 3 a vyššie. Množstvo testerov v snahe zlepšiť túto oblasť začalo vyvíjať príbuzné modely špeciálne pre testovanie. Ťažkosti s implementáciou a používaním CMM [1] umožnili vzniknúť dvom ďalším procesným modelom: TSP (Team Software Process) a (Personal Software Process). CMM a príbuzné modely sa stále vyvíjajú. SEI (Software Engineering Institute) nedávno uviedol CMMI (CMM Integration) a PCMM (People CMM) modely, ktoré sa pokúšajú rozšíriť aplikovateľnosť CMM koncepcií na všetky inžinierske disciplíny.

## Záver

V minulosti som pracoval v IT firme, ktorá nepoužívala žiadny štandard na zabezpečenie kvality softvéru. Podľa CMM teda dosahovala Úroveň 1. Procesy boli neprehľadné a neefektívne. To malo za následok zvýšené náklady na vývoj. V súčasnosti pracujem v IT firme, ktorá má už 4 roky akreditáciu na CMM Úroveň 2 a čoskoro by mala dosiahnuť Úroveň 3. Všetky procesy sú jasne definované a zdokumentované. Síce to vyžaduje od každého člena tímu zvýšene úsilie, ale to sa niekoľkonásobne vracia vo forme kvalitnejšieho softvéru a vyššej spokojnosti zákazníkov.

Odpoveď na otázku, či je manažment kvality dôležitý, znie jednoznačne áno a každá IT organizácia a firma, ktorá chce obstáť na trhu v konkurenčnom boji, by mala implementovať niektorý zo štandardov na zabezpečenie kvality softvéru.

## Použitá literatúra

1. Evans, I.: *Achieving Software Quality through Teamwork*. Artech House, Norwood, 2004.
2. Gill, N.S.: Factor affecting effective software quality management revisited. In: *ACM SIGSOFT Software Engineering Notes*, Volume 30, Issue 2, ACM Press, New York (2005).
3. Jalote, P.: *Software Project Management in Practice*. Addison Wesley, Boston, 2002.

## Annotation

*Quality management and the effect on the project result*

Developing a good software system is a very complex task. In order to produce a good software product, several measures for software quality attributes need to be taken into account. System complexity measurement plays a vital role in controlling and managing software quality because it generally affects the software quality attributes like software reliability, software testability and software maintainability. Thus, software quality assurance (SQA) needs to be addressed keeping in view the new strategies, tool, methodologies and techniques applicable to software development life cycle.
This essay is a survey study. Its main approach is to provide a compact view on the quality management in the software development process. The author answers the fundamental questions, what is quality management and why it is important. The author then compares existing models and standards for software quality assurance. In the conclusion, he describes the effect on the project result based on his practical experience.

# Manažment vzťahov so zákazníkmi

## Môže softvér zlepšiť obchodné vzťahy?

MICHAL HABALA

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
misko@post.cz

**Abstrakt.** Voľba správnej marketingovej stratégie je základom úspechu každého podniku bez ohľadu na oblasť pôsobnosti. Pri vytváraní tej správnej stratégie musí podnik zohľadniť všetky aspekty neustále sa meniaceho trhového prostredia a kľúčom k úspechu je získanie nových a udržanie existujúcich zákazníkov. Do centra pozornosti sa dostáva zákazník ako jednotlivec so svojimi konkrétnymi požiadavkami a tým transformuje masový marketing na marketing individuálnych vzťahov so zákazníkmi (angl. One-to-one marketing). Vo svetle nového prístupu ku zákazníkovi vystupuje manažment vzťahov so zákazníkmi (angl. CRM – Customer Relationship Management) ako nová perspektíva podnikania založená na poznaní, že zdrojom príjmov spoločnosti je zákazník a nie produkt. Na zlepšenie efektivity zložitého prístupu ku zákazníkovi sú vo svete IT vyvíjané špecializované systémy na manažovanie vzťahov s jednotlivými zákazníkmi a partnermi spoločnosti, ktorých nasadenie sľubuje značné prínosy vo forme zvýšenia obratu a lepšieho uspokojenia širšieho okruhu zákazníkov.

## Úvod

Základom každého obchodu sú dve strany: obchodník, ktorý ponúka tovar alebo poskytuje služby a zákazník, ktorý kupuje. Na zabezpečenie stálych príjmov a na ich zvyšovanie je na strane obchodníka (podniku) potrebné neustále predávať svoje produkty, čo zahŕňa potrebu prinášať na trh nové, najlepšie bezkonkurenčné výrobky a služby a získavať nových zákazníkov z rôznych častí trhu ako aj udržanie si spokojných a stálych odberateľov. Podľa môjho názoru je výhodnejšie prijať stratégiu udržania si čo najširšieho okruhu zákazníkov a postupného rozširovania o nových zákazníkov. Pri vývoji softvérových produktov (rovnako ako v ostatných odvetviach) sa teda dostáva do popredia filozofia neustáleho uspokojovania už získaných a obslúžených zákazníkov. Táto filozofia vyžaduje stále zdokonaľovanie komunikačných kanálov na prenos informácií a požiadaviek od zákazníka a novým trendom sa stáva dolovanie informácií o potrebách zákazníka aj z na prvý pohľad nepodstatných informácií.

*Manažment v softvérovom inžinierstve, december 2005, s. 20-27.*

Firmy chcú, aby sa zákazníci stali ich najlepšími priateľmi. Bolo to tak vždy, ale teraz sú zúfalí, keď im musia preukázať svoju vďačnosť. Dnešní zákazníci sú totiž nestáli. Môžu využiť internet na porovnanie cien a skontrolovanie ponuky. Môžu nakupovať tovar a služby online, pomocou telefónu či osobne a vybrať si medzi rôznymi formami prístupu podľa svojho želania. V digitálnom veku očakávajú lepšie, rýchlejšie a priateľskejšie služby a stávajú sa netrpezlivými, keď to chýba.

S rozvojom mobilného obchodu sú užívatelia náročnejší. Pod tlakom potreby odpovedať, spoločnosti sa čoraz viac obracajú na mix podnikateľskej stratégie, spracovania a technológie známej ako Manažment vzťahov so zákazníkmi (CRM). Pred niekoľkými rokmi bolo CRM novinkou. Jeho základné princípy však boli známe už niekoľko desaťročí. V čom je teda CRM skutočne nové? Tradičná marketingová teória opisuje postupný vývoj od produktovo orientovaných marketingových stratégií cez odbytovo orientované až po trhovo alebo zákaznícky orientované marketingové stratégie.

Zákaznícky orientovaná firma sa vyznačuje tým, že sa najskôr snaží zistiť, čo potrebujú súčasní a potenciálni zákazníci, a všetky vnútorné procesy a štruktúry potom orientuje na naplnenie týchto potrieb. Jednoducho povedané, nejde už o to, získať produkt a potom ho umiestniť na trh a snažiť sa preň získať odberateľov, ale ide o to, aby si firma najprv zistila, čo vlastne môžu jej stáli i potenciálni zákazníci potrebovať, potom nájde potrebný produkt a ponúkne im ho [3].

## Koncepcia „získať a udržať“

Manažment vzťahov so zákazníkmi je novým prístupom k podnikaniu, ktorý je založený na presvedčení, že hlavným zdrojom ziskov pre podnik je zákazník, nie produkt. Možno ho definovať ako neustále sa opakujúci proces zameraný na lepšie pochopenie a predpokladanie potrieb, správania a ziskovosti zákazníkov s niekoľkými parciálnymi cieľmi [5]:

- definovanie stratégie podniku vzhľadom na vzťahy so zákazníkom
- optimalizácia potrebných zdrojov pre efektívne podnikanie
- kvalita služieb pri každom kontakte s existujúcim alebo potenciálnym zákazníkom

Podnik, ktorý sa rozhodol dať sa na cestu manažovania vzťahov so zákazníkmi sa musí zmieriť s neustálym prehodnocovaním všetkých svojich aktivít, procesov a služieb z pohľadu potrieb a želaní najhodnotnejších zákazníkov s cieľom udržať ich a získať si ich vernosť. S ohľadom na to, že mnoho spoločností sa už tradične orientuje na svoje produkty, vyžaduje uplatnenie manažmentu vzťahov so zákazníkmi často kompletnú zmenu podnikovej kultúry, prehodnotenie všetkých podnikových štruktúr, aktivít a procesov.

CRM je teda nová marketingová stratégia, v ktorej centre sa nachádza zákazník. Umožňuje firmám pritiahnuť, obslúžiť a udržať si najlepších, teda najziskovejšie zákazníkov a súčasne robiť všetko na získanie nových zákazníkov. Ide vlastne o partnerstvo, kde vy ako obchodník, profitujete zo ziskov zákazníka a preto sa snažíte

o maximalizáciu jeho ziskov, čo spôsobí vyššie uspokojenie jeho potrieb, ktoré sa prejaví väčšou lojálnosťou voči vám. Nejde o jednorázový, ale o dlhodobý vzťah, ktorý je jedinečný. V praxi to tiež znamená, že sa nenaháňame len za zvyšovaním svojho podielu na trhu získavaním ďalších zákazníkov, čo je mimoriadne nákladné až zničujúce. Ale, že tým zákazníkom, ktorých už máme, ponúkame stále širšie a kvalitnejšie doplnkové produkty či nové a ziskovejšie produkty. Preto ani nie je možné, aby konkurencia „odkopírovala" tento vzťah, ako to môže urobiť napríklad tým, že ponúkne inú variantu vášho nového produktu, či dokonca od výrobcu nový produkt, ktorý ste objavili, jednoducho tiež odkúpi [3].

## Starostlivosť o zákazníka počas jeho životného cyklu

Na preplnenom konkurenčnom trhu, kde zákazníci bez problémov prechádzajú ku konkurenčným firmám, môže byť snaha o získanie a udržanie vysokého trhového podielu veľmi nákladná až sebazničujúca [5].

Manažment vzťahov so zákazníkmi sa preto zameriava nie na zvyšovanie podielu na trhu, ale podielu na zákazníkoch. Myslí sa tým snaha predať zákazníkovi čo najviac vysoko ziskových produktov na báze individuálnych vzťahov počas doby jeho životnosti.

Dobu životnosti zákazníka môžeme definovať ako čas od prvého kontaktu so zákazníkom, predaj produktov, ponúknutie a predaj dodatkových služieb a výhod až po ukončenie obchodných vzťahov. Životný cyklus zákazníka potom obsahuje viacero fáz, ktorými prechádzajú vzťahy zákazník-obchodník (v prípade softvérových produktov to môžu byť dokonca vzťahy zákazník-vývojár) podľa toho, či sa jedná o nového zákazníka, opakované komunikovanie so zákazníkom o predaji produktov alebo o udržiavanie vzťahov a technickú podporu zákazníka.

Manažment vzťahov so zákazníkmi predstavuje súhrn aktivít, ktoré vytvárajú ucelený proces zameraný na udržanie existujúcich zákazníkov, na predaj nových produktov existujúcim zákazníkom alebo na predaj ziskovejších produktov existujúcim zákazníkom.

Pre manažment vzťahov so zákazníkmi musí spoločnosť disponovať veľkým množstvom informácií o svojich zákazníkoch v pravý čas a na správnom mieste. Základom pre získanie zodpovedajúcich informácií sú presné a úplné dáta o zákazníkoch a vhodná technológia pre spracovanie veľkej masy dát na potrebné informácie. Na ich základe potom manažment spoločnosti definuje zákaznícky orientovanú a diferencovanú stratégiu.

Prvkami, ktoré determinujú manažment vzťahov so zákazníkmi sú:

- zákaznícky orientovaná stratégia
- zákaznícky diferencovaná stratégia
- zákaznícky orientované podnikové procesy
- technológia umožňujúca riadenie vzťahov so zákazníkmi

Zákaznícky orientovaná stratégia umožňuje pochopiť a optimalizovať všetky procesy zamerané na zákazníka. Pre úspešné riadenie vzťahov so zákazníkmi je nevyhnutné formulovať nielen zákaznícky orientovanú ale aj zákaznícky diferencovanú stratégiu. Táto potreba je vyvolaná tým, že zákazníci každej spoločnosti svojimi názormi a požiadavkami tvoria často veľmi rôznorodú skupinu. Rozdielne programy a stratégie by mali byť definované pre oslovenie zákazníkov podľa toho, v akej fáze vzťahu k spoločnosti sa nachádza. Iný program je potrebný pre získanie nových zákazníkov, iný pre zvyšovanie lojality a ziskovosti existujúcich zákazníkov a iný pre udržanie a znovu získanie zákazníkov. Za najdôležitejšie faktory sa považujú vernosť a udržanie zákazníkov, prostredníctvom ktorých sa buduje a zvyšuje ich ziskovosť

## Základ je v pochopení definície konkrétneho zákazníka

Základom pre definovanie optimálnej zákaznícky orientovanej a diferencovanej stratégie a pre vybudovanie dlhodobého ziskového vzťahu s konkrétnymi zákazníkmi je nájdenie optimálnej rovnováhy medzi tromi kľúčovými charakteristikami vzťahu zákazník – spoločnosť [5]:

- potreby a požiadavky zákazníka
- ziskovosť zákazníka
- rizikovosť zákazníka

Poznanie týchto charakteristík vychádza z jasnej definície zákazníka. Tá následne umožní zamestnancom spoločnosti pochopiť všetky stránky zákazníkovho správania v každom okamihu kontaktu s ním. Prvoradé je pochopenie charakteristík (demografických, ekonomických, sociálnych) svojich zákazníkov, príčin ich správania, faktorov, ktoré podmieňujú ich súčasnú a budúcu ziskovosť.

Cieľom implementácie zákaznícky diferencovanej stratégie je riadenie a optimalizácia celopodnikového procesu tak, aby zákazník v každom okamihu dostal jasnú správu o spoločnosti a aby boli presne splnené jeho požiadavky. Manažment vzťahov so zákazníkmi umožňuje lepšie pochopenie zákazníka ako individuality a priblíženie sa k jeho potrebám v každom bode kontaktu s ním.

Užitočné informácie o svojich zákazníkoch môže spoločnosť získať aj z externých zdrojov, ako sú cielené ankety a demografické výskumy. Na strategickej úrovni je potom nevyhnutné realizovať proces neustáleho zhromažďovania všetkých dostupných dát o zákazníkoch a ich preskúmavanie prostredníctvom analýz a reportingu. Dáta sa tak postupne menia na strategické poznatky o zákazníkoch. Ich zhromažďovaním sa vytvára strategický systém pre podporu rozhodovania.

Pokiaľ sme schopní vyhovieť nášmu zákazníkovi je to významný úspech. Rovnako dôležité je však vedieť, ako sme mu vyhoveli, čím a prečo. Tieto informácie budú základom pri ďalšom rozhodovaní [1].

## Všeliek alebo obtiažne východisko pre zvýšenie efektivity?

Dnes sú podnikové riešenia na báze CRM prístupu pomerne bežné u veľkých firiem, ale stále je možné sa stretnúť s problémami pri zavádzaní a využívaní takto orientovaných podnikových systémov, ktoré pramenia z nepochopenia celkovej stratégie orientácie na zákazníka alebo z nesprávneho používania systémov. V tejto časti uvádzam niekoľko, podľa môjho názoru dôležitých, vyjadrení popredných členov konzultačných spoločností v oblasti manažmentu vzťahov so zákazníkmi.

Riešenia pre riadenie vzťahu so zákazníkom (CRM) rýchlo dosiahli bezprecedentný úspech na oddeleniach predaja, marketingu a služieb zákazníkovi. Nedostatočne definované stratégie CRM v podniku však môžu mať negatívny dopad na efektivitu ich nasadzovania [4].

CRM sa zoširoka označuje za dôležitú cestu zvyšovania tržieb. „CRM je jasne najhorúcejšie východisko dneška," hovorí Chris Gant, partner v KPMG Consulting. Avšak, ako zistili mnohí sklamaní podnikatelia, CRM nie je všeliek. Nie je to jednoducho otázka nakupovania softvéru a postupov na zdokonalenie vzťahov so zákazníkmi a na dodanie im odvahy, aby viac nakupovali. Informačné technológie sú určite životne dôležitou časťou CRM, ale sú nepoužiteľné bez základného pochopenia zásadných podnikateľských nástrojov a metód. "CRM nie je technologický problém, ale je ho veľmi ťažké riešiť bez technológie," povedala Jennifer Kirkbyová, analytička u Gartnera [3].

Spoločnosť IDL na začiatku roku 2001 vykonala medzi európskymi organizáciami z oblasti bankovníctva, poisťovníctva a telekomunikácií výskum European CRM Audit 2001 [2]. Podľa výskumu až 50% európskych organizácií je nespokojných s ich existujúcim CRM systémom. Hlavný dôraz organizácie kladú na rozšírenie možností analýzy profitability, pričom až 93% zákazníkov to pokladá za kľúč k úspechu CRM. Iba veľmi malá časť organizácií prešla k plne zákaznícky orientovanému obchodnému modelu, 53% európskych organizácií analyzuje ziskovosť iba podľa produktu a 25% organizácií nie je schopných analyzovať ziskovosť žiadnym spôsobom.

Presun dôrazu od profitability produktov na profitabilitu zákazníkov nie je prekvapujúci. Je kľúčom k úspechu v tvrdej konkurencii. Napríklad aj menej ziskový produkt môže byť kľúčom k zákazníckemu segmentu, ktorý môže byť vysoko profitabilný.

Zatiaľ čo navonok mnohé spoločnosti teda implementovali technológiu, pri pohľade zvnútra zisťujeme, že riešenia CRM sú vnútorne využívané len nedostatočne. "Zatiaľ čo miera akceptácie a implementácie softvérových súborov CRM na podnikovej úrovni rastie, to isté nemožno tvrdiť o jej prijímaní zamestnancami. Efektívna implementácia v skutočnosti nesúvisí s technológiou, ale skôr s akceptáciou zo strany zamestnancov," hovorí priemyselná analytička u Frost & Sullivan, Katherine Shariq. Prví užívatelia sa na CRM pozerali ako na niečo, čo vyrieši všetky ich problémy. To sa však nestalo, pretože technológia nebola skombinovaná s jasnou, celopodnikovou stratégiou CRM. "Spoločnosti si naozaj vytvorili podporné interné procesy. Veľká časť úspechu technológie CRM súvisí skôr so zmenou in-house systémov a nie až tak veľmi s implementáciou technológie," tvrdí Shariq [4]. "Z

oddelení, ktoré si zadefinujú svoje ciele na začiatku a potom implementujú aplikácie sa stávajú spokojní a úspešní užívatelia - v porovnaní s účastníkmi, ktorí očakávajú, že technológia sama o sebe vyrieši všetky ich problémy," uzatvára Shariq.

## Faktory úspechu CRM

Pre úspešné nasadenie a profitovanie z CRM systémov je potrebné zvládnuť teoretické podklady tejto rozsiahlej filozofie. Ako som spomínal v predchádzajúcej časti, nepochopenie alebo nie komplexné využívanie týchto systémov má automaticky za následok zlyhanie podpory manažmentu vzťahov a býva často spoločnosťami vyhodnotené ako zbytočná investícia. Problémom je, že väčšina systémov CRM v prvom rade ráta s užívateľmi, ktorí sú v priamom osobnom styku so zákazníkom a systematicky vytvárajú databázu znalostí o zákazníkovi a jeho potrebách. Keď tak užívatelia nerobia, lebo je to príliš ťažké, príliš nepohodlné alebo to prináša minimálne výhody, mnohé potom zlyhajú. PA Consulting Group v práci s klientmi ukázala päť kľúčových faktorov úspechu, ktoré prispievajú k efektívnosti implementácie CRM:

- Choď na zákazníka s nevtieravou technologickou pomocou. Predstav si predavača, ktorý má len desaťminútovú príležitosť na ovplyvnenie zaneprázdneného profesionála. PDA je na to nevtieravý nástroj (na rozdiel od laptopu), ktorý prináša kľúčové informácie v rýchlej a efektívnej podobe. Tento princíp použite pre všetky obchodné situácie, vo všetkých oblastiach a nielen prostredí, kde zohráva svoju úlohu časový tlak
- Doprav jednoduchý pohľad zákazníka do hodnotového bodu. Predstav si účtovného manažéra, ktorý má tesne pred alebo počas stretnutia so záujemcom alebo zákazníkom len necelú minútu času na predchádzajúcu informáciu o kontakte, súčasný stav objednávky, otázky a sťažnosti a na vzťahujúce sa podnikové informácie
- Na požiadavku zákazníka odpovedaj okamžite. Predstav si, že musíš byť schopný terminovať servisné návštevy okamžite ako prídu a doručiť servisným manažérom informáciu, ktorú potrebujú na skompletizovanie hovoru (ako sú predchádzajúce problémy, technické špecifiká, burzové informácie), rovnako ako minimalizovať manuálne chyby v nasledujúcom fakturačnom procese
- Otvor nový kanál k svojmu zákazníkovi. Prečo neotvoriť tento nový kanál vybudovaním podpory mobilnej práce? Stále viac zákazníkov bude používať mobilnú technológiu od cieleného marketingu po prúd objednávok, dodávok a otázok
- Maximalizovať príležitosti vybudovať vzťahy so zákazníkom. Vaši ľudia v teréne musia mať všetky možnosti zamerať sa na to, kde prinesú najväčšiu hodnotu, teda na priame stretnutia so zákazníkmi. Mobilná technológia znižuje režijné náklady na administratívu, oneskorovanie korešpondencie a

chyby. Spolu s online prístupmi do ,,kancelárskeho" prostredia urobí vašu mobilnú pracovnú silu produktívnejšou

## Záver

Zákazník bol vždy dôležitý pre podnik a podnikanie, bez získania správneho zákazníka nie je možné podnik ďalej rozvíjať a v nekonečnom konkurenčnom boji by nebolo možné prežiť. Preto vzťahy medzi zákazníkom a obchodníkom, v prípade softvérových firiem medzi zákazníkom a vývojárom, je potrebné neustále rozvíjať a zdokonaľovať v prospech spokojnosti zákazníka. V dnešnej dobe sú obchodné vzťahy na veľmi vysokej úrovni a pre ich ďalšie zlepšenie bolo potrebné zaviesť ďalšiu rovinu v prístupe k uspokojeniu zákazníckym nárokov a potrieb. Správny smer sa ukázal v podobe podrobného skúmania požiadaviek a analýzy potrieb zákazníka, z ktorých je možné modernými metódami vyvodzovať závery do budúcnosti a tak prísť na trh s produktmi, ktoré zákazníci ešte len budú potrebovať. Ak si predstavíme komplexnosť dnešného trhu, je pochopiteľné, že pri aplikovaní takýchto analýz sú potrebné najnovšie technológie a moderné prístupy komunikácie so zákazníkom. Pre tieto dôvody sa v posledných rokoch vo veľkej miere začali používať systémy na podporu manažmentu vzťahov so zákazníkmi. Použitie týchto systémov je pomerne zložité a vyžaduje komplexné podnikové riešenia kombinované s celkovou orientáciou podnikových procesov na zákazníka. Podľa môjho názoru je využívanie týchto sofistikovaných systémov jedinou cestou pre veľké podniky ako si udržať stálych zákazníkov, získať nových a udržať si tak postavenie na trhu a konkurenčné výhody.

## Použitá literatúra

1. Horváth, R.: Silno integrované CRM dobrou voľbou. *proManager – online časopis o informačných a komunikačných technológiách pre manažérov a IT pracovníkov*, www.manager.sk/clanok.sp?id=1614
2. Lencz, T.: CRM-prísľub konkurenčnej výhody. *proManager – online časopis o informačných a komunikačných technológiách pre manažérov a IT pracovníkov*, www.manager.sk/clanok.sp?id=1483
3. Novák, M.: Nové cesty ako vyhrať nad nestálym zákazníkom. *proManager – online časopis o informačných a komunikačných technológiách pre manažérov a IT pracovníkov*, www.manager.sk/clanok.sp?id=1608
4. *proManager – online časopis o informačných a komunikačných technológiách pre manažérov a IT pracovníkov*, www.manager.sk/clanok.sp?id=1777
5. Uramová, Ľ., Komár, M.: Manažment vzťahov so zákazníkmi. *proManager – online časopis o informačných a komunikačných technológiách pre manažérov a IT pracovníkov*, www.manager.sk/clanok.sp?id=160

## Annotation

*Customer relationship management*

Selection of the right marketing strategy is the cornerstone of success of each enterprise, regardless of its target market sector. To create the correct strategy the enterprise has to take into consideration all aspects of constantly evolving market environment, with the key to success being extension and preservation of its customer base. In the center of this effort is the customer as a person, with his/her concrete demands, which transforms mass marketing into the marketing of individual customer relationships – one-to-one marketing. This highlights Customer Relationship Management as a new perspective of business, based on the acknowledgement of the customer, not the product as the source of the enterprise's income. New specialized systems intended to increase efficiency of the customer relationship process are developed each day in the IT sector and their deployment offers significant rewards in the form of increased incomes and the satisfaction of a broader customer base.

# Manažment softvérového projektu a organizácia softvérových tímov

KRISTIÁN SZOBI

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
szobi@chello.sk

**Abstrakt.** Pri vývoji softvéru hrá dôležitú úlohu zloženie a spôsob organizácie tímov. Táto esej sa zaoberá pohľadom na organizáciu softvérových tímov zo sociálneho pohľadu. V eseji približujem základné typy sociálnych štruktúr, popisujem ich prístup k softvérového procesu i projektu a výhody i nevýhody jednotlivých typov. Na záver spomínam typ používaný v praxi, napríklad firmou Microsoft, v ktorej som pracoval.

## Úvod

Organizácia softvérových tímov tvorí jednu z hlavných častí pri manažovaní softvérového projektu. Štruktúra organizácie závisí od viacerých faktorov. Spomeniem typ vyvíjaného softvéru, jeho zložitosť a v neposlednom rade aj zloženie samotných tímov.

Pri vývoji softvéru je veľmi dôležitý aj sociálny aspekt [6]. Rozoznávajú sa tri základné sociálne štruktúry. Pre každý z nich zhrniem a porovnám vlastnosti z pohľadu konceptualizácie úloh, metód a nástrojov pri vývoji softvéru. Uvediem aj stručný popis hybridných modelov, ktoré sa používajú napríklad v spoločnosti Microsoft alebo v iných spoločnostiach predávajúcich softvér.

## Tri typy štruktúr vo vývojových tímoch

| Typ | Definícia |
|---|---|
| Sekvenčný | Softvérový vývoj je produkčná snaha založená na lineárnej množine diskrétnych úloh. Ľudia pracujú v špeciálnych funkciách s formalizovanou interakciou na báze funkcií. Ľudia sú hodnotení za ich špecializované schopnosti |
| Skupinový | Na softvérový vývoj sa pozerá ako na kombináciu vývoja a produkcie, kde množina diskrétnych úloh je opakovaná až pokiaľ produkt nie je hotový. Vývojári sú organizovaní do vzájomne závislých skupín a sú |

*Manažment v softvérovom inžinierstve, december 2005, s. 28-33.*

| | |
|---|---|
| | hodnotení za ich špeciálne schopnosti a za schopnosť pracovať s druhými. |
| Sieťový | Na softvérový vývoj sa pozerá ako na proces stáleho vývoja so špecifickým zameraním na výsledok/produkt. Úlohy nie sú sekvenčné a sú zviazané s jednotlivcami (alebo malými skupinami), ktorých účasť je založená na interakcii. Členovia skupiny sú hodnotení za to, čo dokážu vyprodukovať. Tento prístup implikuje zložitú sieť väzieb medzi ľuďmi a centralizovaným manažmentom. |

**Tab. 1** Popis základných sociálnych typov

Existujú tri základné typy štruktúr vo vývojových softvérových tímoch: sekvenčný, skupinový a sieťový (ich bližší popis je v tabuľke č.1). Tieto štruktúry prezentujú idealizované pohľady [6]. Použijem ich, aby sme lepšie pochopili problémy hybridných modelov softvérového vývoja – modelov, kde sú skombinované vlastnosti viacerých typov.

## Sekvenčný typ

Sociálna štruktúra zapuzdrená v sekvenčnom type je priama a predpokladá známe a predšpecifikované požiadavky na vyžadované úlohy. Sociálnu štruktúru môžeme vidieť ako nezávislú od väčšieho sociálneho a organizačného kontextu. Aj keď je táto štruktúra hierarchická, potreba diskusie je malá. V rámci projektu má každá osoba úlohu, ktorá je diskrétna a špecializovaná [6]. Naviac, predšpecifikované požiadavky na úlohy ďalej implikujú predpísaný pohľad na produkčný proces. To znamená, že sekvenčný typ je podporovaný tvrdením, že dobrý proces vedie aj k dobrému produktu.

Základom sociálnych interakcií v sekvenčnom type je koncept riadenia. Z toho vyplýva, že interakcie medzi ľuďmi sú sledované len z pohľadu funkcie práce, ktorú vykonávajú. Tento pohľad umožňuje výmenu ktoréhokoľvek človeka iným s tou istou úrovňou schopností.

Dôraz v práci zahrňuje potrebné informácie na pracovný produkt a/alebo dokumenty na splnenie úlohy. Môžeme teda povedať, že potrebná komunikácia v rámci tímu i komunikácia medzi tímami môže byť definovaná, formalizovaná a pravdepodobne aj automatizovaná [6]. Sekvenčný typ by teda profitoval z automatizovaných činností. Orientácia riadenia, formalizované interakcie medzi členmi tímu a zameranie na automatizované činnosti znamenajú, že nie je takmer vôbec potrebné, aby vznikli silné putá medzi členmi tímu [6]. Príkladom sekvenčného typu je napríklad tradičný vodopádový model.

Na základe úrovne špecializácie a oddelenia úloh, práca pri sekvenčnom type sa javí ako rutina. To môže spôsobiť, že členovia tímu len veľmi ťažko vidia hodnotu ich individuálnej práce v rámci celku. Oddelenie úloh môže tiež viesť k ohraničenej interakcii s inými členmi tímu a zmenšiť pravdepodobnosť, že sa vytvorí súdržnosť

softvérového tímu. Typické prístupy na zjemnenie týchto prípadov sú „cross-training“ a väčšie zasahovanie manažmentu napríklad uskutočňovaním formálnych stretnutí [5].

V sekvenčnom type spôsobujú orientácia na proces a špecializácia úloh, že individuálny člen môže ľahko ignorovať prácu týkajúcu sa viacerých úloh (ako napríklad dokumentácia). Je to spôsobené tým, že slabý výkon sa ukáže v inej časti procesu a nie v tej, v ktorej bol zavedený. Napríklad zlé požiadavky neovplyvnia priamo analytika a ani zlý kód priamo neovplyvní programátora.

Existujú viaceré techniky na zmiernenie tohto nedostatku - napríklad revízie a kontrolné zoznamy (check lists). Použitie týchto techník má často dva dôsledky:

- Priestor pre zásah manažmentu
- Prerozdelenie členov tímu aby boli súčasťou aj inej úlohy

Pretože je sekvenčný typ založený na rutine, cieľom je často automatizácia. Tento trend môžeme vidieť vo vývoji integrovaných vývojových prostredí a CASE systémov [6].

## Skupinový typ

Sociálna štruktúra v skupinovom type odráža interaktívnu snahu. V tomto type sú procesy vývoja softvéru založené na množine preddefinovaných úloh avšak do úvahy sa berú aj individuálne schopnosti a slabé stránky členov tímu. Na úlohy, ktoré sú často sekvenčné, sa môžeme pozerať iteratívne a explicitná pozornosť je zameraná na zlepšovanie procesu členmi skupiny [6].

Skupinový typ v jeho iteratívnej povahe taktiež explicitne rozlišuje, že softvérový vývoj a produkcia sú často úzko spojené [6]. Preto je skupinový typ normatívny. Navyše, skupinový typ navrhuje voľnú hranicu medzi tímom a sociálnym kontextom a tvrdí, že prelínanie hraníc je dôležitou časťou vývoja. Hlavná myšlienka však stále odzrkadľuje prístup, v ktorom najdôležitejší je proces rovnako ako v sekvenčnom type.

Skupinový typ softvérového vývoja vytvára explicitnú požiadavku na sociálnu interakciu medzi členmi skupiny [6]. Sociálne štruktúry sú orientované na riešenie nevyhnutných konfliktov, ktoré vznikajú pri spolupráci. To znamená, že tu existuje potenciál pre čiastočné automatizovanie produkčných úloh. Taktiež sú potrebné aj dodatočné nástroje a metódy, ktoré priamo podporujú a/alebo umožňujú spoluprácu medzi členmi tímu. Príkladom tohto typu môže byť špirálový prístup.

Skupinový typ vníma dôležitosť interakcie medzi členmi tímu a snaží sa medzi nimi dosiahnuť zhodu. Skombinovaním s cyklickou a integrovanou povahou produkčných úloh sa navrhuje, aby členovia tímu boli vyškolení a podporovaní v procese učenia sa skupiny a manažmente konfliktov [5]. To znamená, že v skupinovom type je úlohou zlepšiť schopnosti člena tímu v tímovej spolupráci.

Z pohľadu skupiny je iteratívna povaha projektu veľmi dôležitá [6]. Ako však tímy vedia kedy prestať iterovať? Môžem s istotou povedať, že najznámejšou formou kontroly iterácie sú peniaze alebo nejaké iné ohraničenie zdrojov. Avšak iná možnosť sa javí napríklad zavedením medzi-iteračných úloh ako sledovanie zmien a verzií, kontrola „vydania“ a plánovanie „vydania“ [5].

Skupinový typ by ocenil nástroje podporujúce spoluprácu v tíme. Skupinový typ taktiež využíva CASE nástroje ale pohľad na ich použitie je iný ako v sekvenčnom type – sústreďuje sa na podporu spolupráce [7].

## Sieťový typ

Sociálna štruktúra v sieťovom type predpokladá kolekciu vzťahov medzi členmi siete. Tieto vzťahy odrážajú frekvenciu a hodnotu zo vzájomných interakcií. Silnejšie väzby vznikajú pri spoločných záujmoch, značnom zdieľaní informácií a interakcií na vyšších stupňoch. Slabšie väzby znázorňujú menej časté interakcie a rôzne formy toku informácií. V sieťovom type sa v procese vývoja zvyčajne objavujú a premietajú sieťové väzby, ktoré sa vyvinuli medzi účastníkmi [6]. Na rozdiel od prvých dvoch typov, prioritu má produkt a samotný proces je až druhoradý. V sieťovom type individuálni členovia a sociálne vzťahy medzi nimi definujú snahu softvérového vývoja. Navyše, táto sieť je plne zahrnutá v širšom sociálnom kontexte, ktorý nemusí byť ohraničený organizačnou alebo geografickou hranicou [6].

Vznikajúci proces je ohraničený úlohami, ale tieto úlohy sú funkciou sociálnych vzťahov, ktoré tvoria sociálnu sieť. Takže, aj keď nie je tento proces centrálny, existuje nejaká forma kontroly verzií, testovania a dokumentácie. Sieťový typ vychádza z tvrdenia, že dôležitým faktorom pri tvorbe dobrého produktu je mať dobrých ľudí [6]. Z tohto prístupu, kde sa dôraz kladie na samotných ľudí, vychádza aj tvrdenie, že je veľmi náročné, možno až nemožné, vymieňať kľúčových členov (člena) siete, ku ktorým vedie veľa spojení v rámci siete. Títo ľudia tvoria spojenia, ktoré vytvárajú samotnú sieť.

Na podporu siete je kritické aby nástroje na vývoj softvéru poskytovali medziprepojenia [5]. Môžeme jednoducho povedať, že nástroj na vývoj softvéru je cenený na základe toho ako pomáha individuálnemu členovi a/alebo ako dobre umožňuje zdieľanie v sieti. Druhým dôsledkom je to, že sieťový typ, ktorý je zároveň vznikajúci aj popisný, znázorňuje vývojový pohľad na softvér. Centralizácia sociálnych štruktúr a interakcia individuálnych členov v sieťovom modeli často spôsobujú, že snaha skupiny je často sporná [5]. Z tejto perspektívy sú interakcie medzi členmi zamerané na vlastnosti produktu, funkcie a akcie. Exituje zopár procedurálnych detailov na riešenie sporov, avšak očakávania medzi členmi sú vyjadrené ako „ukáž a povedz“. To znamená, že riešenie konfliktov je často založené na poskytnutí zdrojových kódov k podnetu, ktorý sa má vyriešiť.

Vývoju softvéru pomocou modelu „tím hlavného programátora“ je jeden z príkladov. Silné väzby sú medzi členmi a hlavným programátorom. Naopak, slabé väzby sú medzi samotnými členmi tímu. Nárast snahy vývoja otvoreného softvéru odráža druhú formu sieťového typu. Sieť má v tomto prípade viacero hlavných ťažísk v závislosti od dôležitosti a viacnásobné väzby medzi mnohými členmi tímu.

V sieťovom type je často náročné oceniť prínos bez zatajenia ďalších spojení. Kvôli vzájomnej závislosti na práci skombinovanou s individualizovanou podstatou spôsobu vykonania práce, vyhodnocovanie prínosu je založený na výsledkoch. Tento spôsob je použitý aj vo firme Microsoft. Prístup založený na výsledkoch si vyžaduje

silný projektový manažment, ktorý je často centralizovaný do jedného alebo viac uzlov siete. Takže sieťový typ rozptyľujúci prácu vlastne koncentruje manažment. Môžeme teda povedať, že aj keď sa projekt riadi jedným človekom (alebo uzlom), tento sa môže meniť ale málokedy sa riadenie zdieľa [4].

Sieťový typ začína s množinou individuálnych ľudí navzájom spojených vyvíjajúcimi sa sociálnymi väzbami. V tomto prostredí môže nastať problém pri opakovateľnosti a stabilite – preto je dôležitá snaha o zaistenie spoločného procesu pre opakované činnosti ako prideľovanie úloh, hlásenie pokroku a pod. Jednou metódou zlepšenie sieťového typu môže byť verejná projektová sledovacia tabuľa [3]. Táto metóda by pomohla udržať a sledovať vzájomné vzťahy medzi členmi siete.

Nástroje pre použitie v sieťovom type musia na základe jeho povahy podporovať znovupoužiteľnosť a interakciu. To znamená z pohľadu siete najmä jednoduché zdieľanie súborov.

## Hybridné modely

V praxi je viac pravdepodobnejší scenár, že tím si adoptuje hybridnú sociálnu štruktúru, ktorá obsahuje prvky viacerých základných typov. Napríklad, Brooks [2] a Baker [1] píšu o snahe firmy IBM pri vývoji systému System 360. Brooks podčiarkuje dôležitosť štruktúry projektového vývoja, zatiaľ čo Baker vysvetľuje fungovanie tímu hlavného programátora. Sekvenčný typ podporovaný Brooksom poskytuje štylizovaný pohľad na vývoj kým Bakerov popis poskytuje náhľad ako hlavný programátor môže vytvoriť hybridnú sociálnu štruktúru.

Jeden úsudok, ktorý z tohto vyplýva je: aby fungoval vývoj softvéru, hybridná sieťová štruktúra vzťahov je potrebná medzi vývojármi [4]. Na základe existujúcich dôkazov si myslím, že je to pravda. Napríklad, Weinberg zaznamenal ako produktivita vývojárov klesla potom, čo boli odstránené automaty na nápoje a sladkosti z oddychovej miestnosti programátorov, pretože následne strávili menej času vzájomnou neformálnou komunikáciou. Odstránením automatov zbavili vývojárov ich závislosti na neformálnej sieti, ktorá vznikala v okolí týchto automatov.

Zaujímavé je porovnanie s vývojom baleného softvéru. Prístupy pri vývoji baleného softvéru sa líšia od tradičného softvérového vývoja. Sám som pracoval vo firme Microsoft. Interakcia medzi vývojármi je väčšia a menej sa spolieha na formálny proces. Naopak, väčší dôraz sa kladie na konštrukciu produktu. Tento dôraz na produkt umožňuje interné vzájomné súperenie medzi jednotlivými vývojármi. V základe, prístup k vývoju vo firme Microsoft je hybridom medzi skupinovým a sieťovým typom.

Na základe tohto prehľadu sa dajú odvodiť ďalšie dva fakty:

- Pre každú snahu v softvérovom vývoji slúži malé množstvo kritických ľudí ako uzly vo vyvíjajúcej sa sieti programátorov
- Akonáhle sa stanú vývojové siete formálne štruktúrované, problémy konfliktov a závislostí sa stanú viac a viac dôležité

## Záver

Sociálny pohľad na softvérový vývoj nám pomáha identifikovať tri základné typy sociálnej organizácie. Ich rozdiely nás vedú k otázke, ktorý prístup je ten najlepší? To je nepochybne empirická i filozofická otázka a ide za hranice tejto eseje. V eseji popisujem vlastnosti jednotlivých typov, možné komplikácie pri ich využití a ich odstránenie. Tak či onak, spomínam hybridný model, ktorý existuje a v ktorom sa miešajú prvky týchto základných typov. Cieľom bolo podať čitateľovi informácie zo sociálneho pohľadu pri výbere typu organizácie softvérových tímov.

## Použitá literatúra

1. Baker, F.: Chief Programmer Team Management of Production Programming. In *IBM Systems Journal*, 11(1), 1972, 56-73.
2. Brooks, F.: *The Mythical Man-Month*, Datamation, 1974. 44-52.
3. Raymond, E.: *The Cathedral and the Bazaar*, Available online at http//www.tuxedo.org/~esr/writings/, 1999.
4. Sawyer, S.: Packaged Software: Implications of the Differences from Custom Approaches to Software Development, *European Journal of Information Systems*, 9 (1), 2000, 47-58.
5. Sawyer, S., Farber, J. and Spillers, R.: Supporting the social processes of software development teams. *Information Technology & People*, 10(1), 1997, 46-62.
6. Sawyer, S.: *Software Developing Teams: Three Archtetypes and their Differences*, 2000.
7. Vessey, I., and Sravanapudi, P.: CASE Tools as Collaborative Support Technologies. In *Communications of the ACM*, 38(1), 1995, 83-95.

### Annotation

*Management of software project and the software team organization*

An important role in the process of software development is the way of organizing software teams. This paper focuses on the software team organization from social perspective. I talk about basic types of social structures and describe their attitudes to both software process and software project, and the cons and pros of the particular type. At the end, I am mentioning a model used in real-world scenario by for example Microsoft corp. in which I used to work.

# Manažment v softvérovom inžinierstve z hľadiska riadenia projektu

# Manažment rizík v softvérovom projekte

IMRICH BALKO

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
balko01@student.fiit.sk

**Abstrakt.** Vysoké nároky na softvér ho robia čím ďalej komplexnejším, čo prináša so sebou väčšie riziko, ktoré môže znamenať neúspešné ukončenie projektu – neskoré odovzdanie, neúplné resp. nesprávne riešenie, atď. Každá spoločnosť sa chce vyhnúť podobným rizikám, preto sa prijímajú opatrenia, ktoré by tieto riziká znižovali v čo najväčšej miere. Takýmto prostriedkom je manažment rizík v softvérových projektoch. Manažment rizík v softvérových projektoch je pomerne mladá disciplína, ktorá znamená prvý krok k úspešnému ukončeniu projektu. Zavedenie manažmentu rizík je nielen moderným, ale aj pre kvalitu a úspech projektu nevyhnutným prístupom riadenia projektu. V tejto práci si rozoberieme čo je to riziko, čo je manažment rizík v softvérovom projekte a aké opatrenia treba vykonať, aby sme sa im vyhli.

## Úvod

Softvér sa stal koncom 20. storočia neodmysliteľnou súčasťou moderného človeka. Jeho používanie sa stalo každodennou rutinou v každom odvetví. Pokrok v technológiách podnietil tiež väčší dopyt po softvéri, ktorý sa čím ďalej stáva viac špecifickejší a tým aj komplexnejší. Predstava zákazníka o konečnom produkte nebýva vždy jasná z hľadiska funkcionality, ale predpokladá, že v danom čase dostane softvér, ktorý bude mať všetky potrebné funkcie. Avšak cesta od požiadavky až po nasadenie je často „dlhá“ a nevyspytateľná. Na vývoji softvérového projektu sa podieľa tím pod vedením projektového manažéra, ktorý je zodpovedný za priebeh celého softvérového projektu. Skúsený projektový manažér pozná potenciálne riziká a snaží sa projekt viesť tak, aby im predišiel. Predísť všetkým rizikám je možné len v ideálnom prípade. V reálnom projekte sú riziká prítomné počas celého životného cyklu. Vznikom problémov vznikajú aj metódy a metodiky ako im predchádzať.

## Čo je manažment rizík a aké sú riziká v softvérovom projekte

Skôr ako si zadefinujeme pojem manažment rizík, potrebujeme vedieť, čo je riziko. Riziko predstavuje *pravdepodobnosť* výskytu ohrozenia (straty, škody). Čiže ak

*Manažment v softvérovom inžinierstve, december 2005, s. 37-43.*

hovoríme o riziku, hovoríme ešte len o pravdepodobnosti, že môže nastať stav ohrozenia.

Pre manažment rizík v softvérovom projekte existuje viacero definícii, ktoré ako uvidíme na (konkrétnych) spomenutých, majú spoločné charakteristiky. Definícia NASA (National Aeronautics and Space Administration): *„Cieľom manažmentu rizík v softvérovom projekte je identifikovať, adresovať a eliminovať rizikové prvky skôr ako ohrozia úspech alebo zvýšia nutnosť prepracovania projektu.“* (NASA, 1999) [1].

Ďalšia definícia hovorí, že manažment rizík je plánované riadenie rizík, ktoré obsahuje dohľad nad úspechom projektu, analýzu potenciálnych rizík a tvorbu rozhodnutí pre prípady potenciálnych rizík.

Ako sme si mohli všimnúť, obe definície spomínajú isté mechanizmy, ktoré sú určené na zmenšovanie a riadenie rizík. Treba si uvedomiť, že aj keď neexistuje žiadna univerzálna definícia, všetky sa odvíjajú do poznatku dvoch hlavných charakteristík rizika:

- *neistota:* (riziková) udalosť sa môže ale nemusí objaviť
- *strata:* (riziková) udalosť má za následok nečakané dôsledky alebo straty

V praxi teória nestačí (ale je nevyhnutná na úspešné zvládnutie projektu). Preto sú pre softvérové projekty určení ľudia, ktorí sú zodpovední za riadenie rizík. Obvykle je to práve projektový manažér, ktorý je zodpovedný za celé riadenie projektu a teda aj za riadenie rizík. Prípadne túto úlohu deleguje zodpovednému členovi projektového tímu.

### Riziká podľa Dr. Barry W. Boehma

Pre lepšie zvládnutie rizík je dobré si brať príklad zo skúseností úspešných projektových manažérov. Jedným z takýchto, ktorý sa podelil so svojimi skúsenosťami, je aj Dr. Barry W. Boehm, ktorý popisuje [3] desať najčastejšie sa vyskytujúcich rizikových prvkov. Tými sú:

1. Deficit ľudských zdrojov
2. Nereálny harmonogram a rozpočet
3. Vývoj zlých funkcií a vlastností
4. Vývoj zlého používateľského rozhrania
5. Gold plating – pridávanie viac do systému ako bolo požadované
6. Prúd pravidelných požiadaviek na zmenu
7. Deficit externe dodávaných komponentov
8. Deficit externe vykonávaných úloh
9. Slabá výkonnosť
10. Vysoké zaťažovanie odborníkov

### Riziká podľa Capersa Jonesa

„Rozsiahle softvérové projekty nikdy nebudú bez nejakých rizík, ale ak budú riziká znižované na akceptovateľnú mieru, bude to dobrý začiatok.“ (Capers Jones, 1998) [1].

Všetky riziká môžeme zhrnúť do nasledovných troch najviac kritických faktorov [1], ktoré sa opätovne vyskytujú vo veľkom počte spoločností:

- softvérové projekty nie sú odhadované a plánované s akceptovateľnou presnosťou
- reportovanie stavu projektu je často nesprávne a klamné
- kvalita softvéru je často veľmi nízka

V prípade výskytu rizikovej udalosti je potrebné zasiahnuť v prospech projektu. Toto majú za úlohu vedúci projektu, zväčša projektoví manažéri. S manažovaním projektu sa objavujú ďalšie tri prídavné rizikové faktory, na ktoré vplýva manažment softvérovej organizácie. Týmito faktormi sú [1]:

- exekutíva často neprijíma presné a opatrné odhady
- exekutíva pridávajú najviac požiadaviek v strede vývoja
- exekutíva často presadzujú rozvrh, ktorý svojím časovým tlakom poškodzuje kvalitu

Ak si to zhrnieme, najväčším nepriateľom v softvérových projektoch je vo všeobecnosti zlý odhad a zlé reportovanie stavu projektu.

**Príklad vzniku rizík v softvérovom projekte**

*Pevný čas a pevná cena*

Neodmysliteľnou súčasťou tvorby ponuky sa stáva klient, ktorý má jasný cieľ [2] – za fixnú cenu, ktorá je čo najnižšia, dostať v požadovanom čase bezchybný a plne funkčný softvér. Výsledný produkt musí byť podľa najnovších technológií a v prípade, že sa (výsledný) projekt neodovzdá v riadnom čase, požaduje čo najvyššie penále.

Tento príklad predstavuje vysoké riziko v každom projekte, keďže čas projektu býva mnohokrát poddimenzovaný z dôvodu prilákania zákazníka.

*Nepresné požiadavky na softvér*

Iným príkladom je zákazník, ktorý síce chce softvér, ktorý mu splní „všetko“, ale nie je schopný úplne špecifikovať jeho požiadavky. Táto situácia predstavuje najväčšie riziko, keďže jej výskytom ohrozuje čas dodania a nutné zmluvné konfrontácie. Zákazník nepresnými požiadavkami získava produkt, ktorý bol ním špecifikovaný, ale nie je podľa jeho predstáv. Tento fakt sa môže odraziť aj na mene spoločnosti, lebo v konečnom dôsledku všetky vzniknuté problémy znáša dodávateľ.

## Ako na riadenie rizík

V prvom rade ja nevyhnutné si uvedomiť, že neexistuje žiadna univerzálna príručka ako riadiť riziká. Každá spoločnosť má inú kultúru, iné návyky. Riziko je len abstraktný pojem, pokým sa nestane problémom. Preto identifikovať a poznať riziká neznamená, že vieme aj riziká riadiť. Riadenie rizík je predovšetkým o skúsenostiach ako v danej situácii (čo najlepšie) konať, aby bol konečný dopad čo najmenší (v ideálnom prípade bez strát).

**Základné kroky riadenia rizík**

Ako každá metodika, tak aj manažment rizík v softvérovom projekte je pre každú metodiku špecifický, ale jeho podstata by mala byť spoločná. Manažment rizík väčšinou pozostáva z nasledujúcich krokov:

- identifikovanie rizík
- pomenovanie akcií
- zoradenie podľa priorít

*Identifikovanie rizík*

Skôr ako sa zavedie proces riadenia rizík, treba vedieť aké rizika vôbec riadiť. Tieto sú úzko špecifické pre každú doménu resp. pre každú spoločnosť. Napríklad riziko pre farmárov môže byť počasie. Ak je počasie nevyhovujúce, môže prísť ku škodám. Pre vývojárov softvéru je to napríklad rýchlosť inovácie a vývoja nových, na trhu neznámych aplikácií. Čo znamená, že ak sa spoločnosť nedrží trendov a zároveň neprichádza na trh s novinkami medzi prvými, vzniká riziko zániku alebo prinajmenšom strate zákazníkov a odlivu financií.

Identifikovanie rizík nebýva často triviálna záležitosť. Zvyšovaním nárokov a zložitosti sa však jednoznačnosť rizík stráca. Preto je potrebné venovať dostatočný čas a pozornosť na ich odhalenie. To znamená nutnosť poznať biznis proces celej spoločnosti (organizácii) a artefaktov, ktoré vstupujú resp. vznikajú pri konkrétnych činnostiach.

*Pomenovanie akcií*

Druhým krokom je pomenovanie akcii, ktoré budú vykonané, ak sa riziko stane skutočnosťou. Často sa stáva, že riziká sú len identifikované a zdokumentované. Tieto riziká sa potom ukážu už ako konkrétny problém, ktorý treba riešiť. Aby sme mali možnosť okamžite a správne riešiť takúto situáciu, potrebujeme okrem identifikácie rizík aj zistiť od vlastníka rizika, akú akciu treba vykonať. Akcie môžeme rozdeliť na [2]:

1. vykonanie akcie nad rizikom – znamená vykonať takú činnosť, ktorá vedie k odstráneniu vzniknutej rizikovej situácie
2. eliminovanie rizika – vykonať takú činnosť, ktorá znižuje prípadne úplne odstraňuje hrozbu rizika. Príkladom môže byť používanie nestabilnej verzie vývojového prostredia. Nahradením stabilnou verziou znižujeme riziko výskytu
3. zanedbanie rizika – nevykonať žiadnu akciu. V konečnom dôsledku vykonanie nejakej akcie je časovo alebo finančne náročnejšie ako zanedbanie rizika. Prípadne vznik rizika nespôsobí žiadne významné škody
4. pozorovanie rizika – sledovanie pravdepodobnosti a dopadu rizika

*Zoradenie podľa priorít*

Tretím krokom je tieto riziká zoradiť podľa priorít. Na efektívne zvládnutie rizík nám nestačí poznať len pravdepodobnosť vzniku a dopad. Tretím potrebným prvkom je časový rámec určujúci poradie, ktoré by jednoznačne jednou hodnotou priradilo toto poradie. Riziko vypočítame [2] nasledovným vzorcom (1):

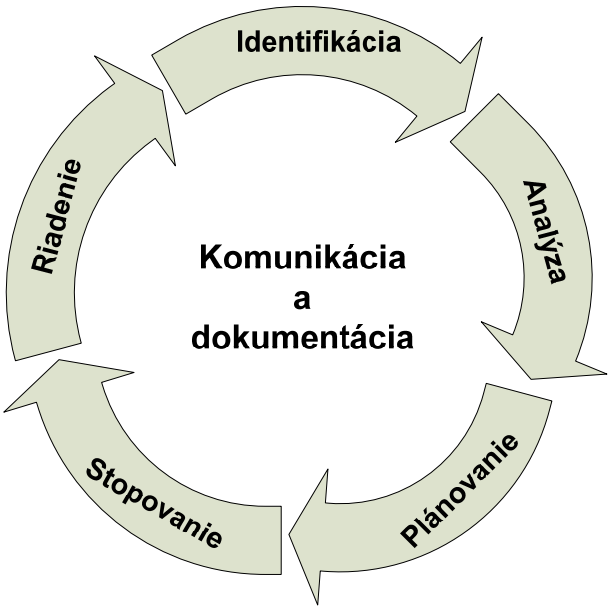
$$\text{Riziko} = \text{Pravdepodobnosť} * \text{Dopad} * \text{Časový rámec} \quad (1)$$

Vynásobenie týchto troch elementov získame číslo priority rizika (Risk Priority Number), ktoré určí konečnú hodnotu priority (časový rámec je len subjektívna hodnota).

**Iteratívny manažment rizík**

Jednou z metód riadenia rizík v softvérovom projekte je iteratívne riadenie rizík [1]. Tu sa proces delí na šesť základných častí. Počas celého životného cyklu musí prebiehať komunikácia a dokumentovanie. Ďalšími časťami, ktoré sa cyklicky opakujú, sú:

- identifikácia
- analýza
- plánovanie
- stopovanie a
- riadenie




**Obr. 1.** Základné funkcie v iteratívnom manažmente rizík.

Na obrázku Obr. 1 môžeme vidieť priebeh procesu.

*Funkcie iteratívneho manažmentu rizík*

Čo je *identifikácia* sme si rozobrali v predchádzajúcej časti. Pomenovanie akcií a určenie priorít rizikám je rozdelené v nasledujúcich štyroch častiach:

- *analýza* – jej funkciou je detailne preskúmať riziko a určiť jeho rozsah ako spolu súvisí s inými rizikami. Analýza pozostáva z určenia dopadu, pravdepodobnosti a časového rámca, z klasifikácie rizík a určenia priority
- *plánovanie* – podľa tejto metodiky sú 4 možnosti plánovania a to skúmať riziko, akceptovať riziko, pozorovať a zmierniť riziko
- *stopovanie* – je proces, pri ktorom sa údaje získavajú resp. zbierajú a reportujú. Účel stopovania je zozbierať presné a relevantné informácie o riziku a následne ich zrozumiteľne prezentovať zodpovedným osobám
- *riadenie* – funkciou je včas informovať, robiť efektívne rozhodnutia ohľadom rizík a upresňovať plánovanie

Počas celého procesu musí byť prítomná *komunikácia* a *dokumentovanie*. Toto sú základné prvky k úspešnému projektu.

## Záver

Manažment rizík v softvérových projektoch je v súčasnosti bezpochyby nevyhnutným opatrením pred vznikom nečakaných udalostí počas celého životného cyklu.

V tejto práci sme si ozrejmili základné pojmy a princípy v manažmente rizík. Ako ukazuje súčasný trend, manažment rizík sa stane súčasťou každej softvérovej spoločnosti, ktorá vyvíja komplexné softvérové riešenia. Komplexnosť projektov prináša so sebou určité riziká, ktoré ak sa identifikujú dopredu, predstavujú menšie straty na zdrojoch. Ukázali sme si ako vo všeobecnosti manažovať riziká a aké činnosti manažovanie prináša. Taktiež sme si ukázali proces iteratívneho manažmentu rizík. Ukázať všeobecný postup je nemožné vzhľadom na špecifické procesy a kultúru v každej spoločnosti. Preto je potrebné, aby si každá spoločnosť špecifikovala vlastné postupy potrebné k manažmentu rizík.

## Použitá literatúra

1. Buttigieg, A.D.: *Risk Management in a Software Development Life Cycle* [online]. Faculty of science, University of Malta. Dostupné na internete: <http://www.cis.um.edu.mt/~abut/>
2. McNair, P.D.: *Controlling Risk* [online]. The Association for Computing Machinery (ACM) – Ubiquity, november 2001. Dostupné na internete: <http://www.acm.org/ubiquity/views/p_mcnair_1.html>

3. MYJ Team: *Introduction to Risk Management* [online]. George Mason University, marec 1997. Apríl 1997. Dostupné na internete: <http://www.baz.com/kjordan/swse625/intro.html>
4. Schoenthaler, F.: *Risk Management in Challenging Business Software Project* [online]. IEEE Computer Society, 2002. Dostupné na internete vo formáte PDF: <http://csdl.computer.org/dl/proceedings/re/2002/1465/00/14650008.pdf>

## Annotation

*Risk Management in Software Project*

High requirements on software cause that software becomes more and more complex. This brings some risks, which can cause unsuccessfully finished projects. Every company's priority is to avoid these risks; therefore they accept actions, which will reduce any risk situations. Risk management is one of possible way to avoid risks. Risk management in software project is relatively new discipline, which can be a first step to success with project. Application of risk management is not only a modern, but even for quality and success necessary approach included in overall software management. In this paper we will analyze what is risk, what is risk management in software project and which actions should be taken to avoid it.

# Odhadovanie v softvérových projektoch

RUDOLF DAČO

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
rudi.sb@post.sk

**Abstrakt.** Efektívne odhadovanie softvérového projektu je jedna z najnáročnejších a najdôležitejších aktivít v procese vývoja softvéru. Správne naplánovanie projektu a jeho kontrola nie je možná bez určenia hodnoverného odhadu. Softvérový priemysel vo všeobecnosti nevenuje dostatočnú pozornosť odhadovaniu a nevyužíva odhady správnym spôsobom. Nesprávne odhadovanie vedie k nesprávnemu využívaniu prostriedkov, čo má za následok neplnenie termínov a plytvanie finančnými prostriedkami. Manažment projektu si často neuvedomuje, že odhadovanie nie je jednoduchý proces, ktorý musí určiť výsledné hodnoty so sto percentnou presnosťou. Odhadovanie je vždy zaťažené určitou chybou a upresňuje sa počas celého priebehu projektu. V tomto príspevku sa pokúšam ukázať, aký je správny postup pri odhadovaní, prečo je dôležité a ako sa robí odhad v projektoch rozdielnej veľkosti.

## Úvod

Základný problém s odhadovaním v softvérových projektov je predovšetkým to, že vo vedení projektu je osoba (manažér projektu), ktorá chce poznať dve čísla: dátum, kedy bude softvér pripravený, a suma akú bude projekt stáť. Tieto čísla chce mať čo najpresnejšie, aby bez akýchkoľvek pochybností vedel rozhodovať o projekte. Bolo by v poriadku, ak by interpretácia týchto čísel bola jasná, ale nanešťastie tomu tak nie je. Osoba, ktorá v konečnom dôsledku rozhodne, či budú do projektu investované peniaze chce počuť, že softvér bude so sto percentnou istotou dokončený pred istým dátumom (ak nie, tak požaduje kompenzáciu) a že so sto percentnou istotou bude projekt stáť menej ako určitá suma. Skutočnosť je však vzdialená od tejto ideálnej predstavy.

Softvérový projekt má množstvo nepredvídateľných faktorov a tie bránia tomu, aby bolo možné jednoducho odhadnúť jednu presnú hodnotu. Preto existujú postupy, ktoré dokážu odhadnúť hodnotu, ktorá sa približuje k tejto presnej hodnote. Tieto postupy sú dnes natoľko sofistikované, že tvoria samostatnú entitu v manažmente projektu nazývanú "Odhady v softvérovom projekte". V mnohých projektoch sa manažmentu odhadov vôbec nevenuje pozornosť. Pokúsim sa ukázať kedy a v akých projektoch je odhadovanie opodstatnené a ako má prebiehať.

*Manažment v softvérovom inžinierstve, december 2005, s. 44-50.*

## Odhadovanie projektu

Odhadovanie softvérového projektu sa vykonáva v štyroch základných krokoch [3]:

1. odhad rozsahu produktu
2. odhad úsilia v hodinách alebo mesiacoch odpracovaných jedným človekom
3. odhad trvania činností v kalendárnych mesiacoch
4. odhad nákladov projektu v peňažnej mene

## Odhad rozsahu

Prvým krokom pri odhadovaní je odhad rozsahu vyvíjaného produktu. Podľa mňa ide zároveň aj o najdôležitejší krok pri odhadovaní. Odhad rozsahu produktu priamo ovplyvňuje ďalšie atribúty projektu: náklady, produktivita, zložitosť, testovateľnosť a pod. Preto si myslím, že aj v tých najmenších projektoch je tento krok neopomenuteľný. Aj pracovník, ktorý pracuje na projekte sám, si musí vedieť odhadnúť rozsah aspoň približne. S rastom rozsahu projektu rastie zároveň aj dôležitosť presnosti tohto odhadu.

Pre správny výsledný odhad je dobré mať čo najviac vstupov. Podľa [3] môžu byť týmito vstupmi: špecifikácia požiadaviek zákazníka, špecifikácia systému, špecifikácia nefunkcionálnych požiadaviek. Odhad rozsahu sa prehodnocuje vždy akonáhle sú nejaké zmeny v týchto vstupoch. Najlepšie sa vykoná odhad pre projekt, ktorý je podobný s už realizovaným projektom. Vtedy postačuje nájsť analógiu medzi novým a starým projektom a percentuálne určiť rozsah nového projektu. Ak nie je možné porovnať nový projekt so starým, vykoná sa nový odhad pomocou niektorej z algoritmických metód: funkčné body (angl. function points – FP), dĺžka textu programu (angl. lines of code - LOC). Takýto odhad je potrebné dobre zdokumentovať, aby bolo možné ho prípadne použiť v ďalšom projekte.

V niektorých projektoch prebieha opačný postup pri odhadovaní. Neodhaduje sa rozsah na základe vstupov, ale definujú sa vstupy na základe rozsahu projektu. Takýmito projektmi sú často projekty na akademickej pôde v rámci vyučovacieho procesu. Tu je presne stanovený rozsah projektu vyplývajúci z trvania projektu a náročnosti projektu. Projekt má väčšinou zákazníkom (vyučujúci) presne definovaný dátum dodania produktu a tento dátum nie je možné zmeniť. Požiadavka môže znieť, aby bol produkt dodaný do troch mesiacov. Ak vývojár (študent) vie koľko času má, jediná vec ktorú môže urobiť je vyjednať množinu funkcionalít, ktoré môže implementovať za požadovaný čas. Vždy je potrebné urobiť viac ako sa dá za stanovený čas stihnúť, preto sú funkcionality vyberané a priorizované tak, aby bol načas odovzdaný súdržný produkt. Preto podľa mňa často dochádza k podceňovaniu dôležitosti odhadovania rozsahu študentmi, ktorí počas štúdia pracovali iba na takýchto projektoch.

## Odhad úsilia

Samotný odhad rozsahu ešte nemá pre prácu na projekte veľký význam. Hovorí síce aký je projekt rozsiahly, no nehovorí aké závery z toho vyplývajú pre projekt. Je nutné tento odhad transformovať do odhadu úsilia, ktoré je potrebné vynaložiť pri tvorbe produktu. Niekedy sa pristupuje priamo už k odhadu úsilia bez toho, aby bol dostatočne analyzovaný rozsah projektu. Môže to viesť ku skresleným výsledkom, preto prvým a najdôležitejším je odhad rozsahu.

Hlavnou úlohou odhadu úsilia je identifikovať a odhadnúť všetky aktivity, ktoré sa budú podieľať na tvorbe produktu. Je možné pritom využiť dáta z minulých projektov. Pritom ale predpokladáme, že:

- organizácia má zdokumentované výsledky z predchádzajúcich projektov
- organizácia riešila aspoň jeden obdobný projekt
- projekt bude mať obdobný životný cyklus, budú použité obdobné implementačné postupy, budú použité obdobné nástroje a tím bude mať obdobné skúsenosti

Ak však organizácia nemá k dispozícii dáta z minulých projektov, prípadne sa nový projekt odlišuje od tých predchádzajúcich, potom je potrebné použiť algoritmický postup pre odhad úsilia (napr. COCOMO). Tieto postupy však rovnako vychádzajú z analýzy už ukončených projektov s tým rozdielom, že tieto projekty nemusia byť podobné s riešeným projektom. Je oveľa presnejšie, ak náš projekt porovnávame s projektom vyvíjaným v rovnakom prostredí (najlepšie v tom istom).

## Odhad rozvrhu

Až keď máme k dispozícii odhad úsilia vychádzajúci z odhadu rozsahu, môžme navrhnúť rozvrh. Po absolvovaní týchto fáz môžme odhadnúť kedy bude produkt hotový a oznámiť túto hodnotu manažérovi projektu.

Podkladom pre vypracovanie rozvrhu je preto odhad úsilia. Predpokladáme, že máme odhad počtu ľudí, ktorí budú pracovať na projekte, čo budú robiť na projekte, kedy začnú pracovať a kedy svoju prácu ukončia. Akonáhle máme množinu týchto informácií, môžme tieto informácie transformovať do postupnosti aktivít v kalendári. Pritom musíme poznať aspoň rámcovo modely životného cyklu softvérového systému, aby sme vedeli, aké činnosti a v akom poradí je potrebné vykonať. Pri tvorbe rozvrhu je opäť možné využiť dáta z minulých projektov. Ak takéto dáta nemáme k dispozícii môžme použiť rôzne metódy a techniky, ktoré boli pre tvorbu rozvrhu navrhnuté (napr. Ganttova schéma, metóda kritickej cesty). Tie nám dajú aspoň čiastočnú predstavu o celkovej dĺžke kalendára.

## Odhad ceny

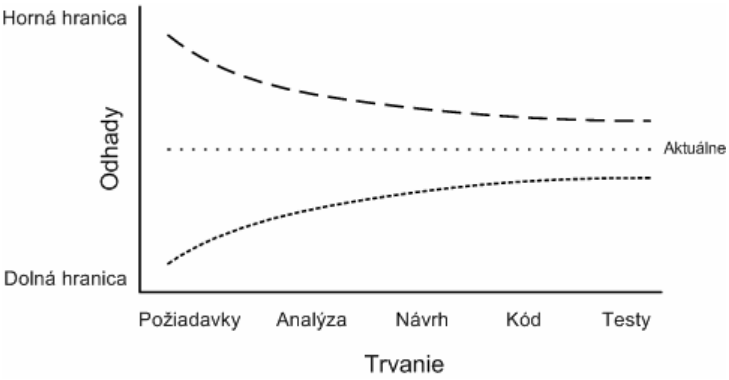
Cena je druhá hodnota, ktorá zaujíma manažéra projektu. Je mnoho faktorov, ktoré vplývajú na určenie ceny, no základný predpoklad je, že máme predstavu o postupnosti aktivít, ktoré treba vykonať a teda, že máme vytvorený rozvrh. Na výslednú cenu má vplyv okrem ceny práce aj cena nakúpeného hardvéru a softvéru, poplatky za cestovanie a komunikačné služby, poplatky za školenia a napríklad aj za prenájom priestorov.

Identifikovať také množstvo nákladov je podľa mňa vo veľkom projekte takmer nemožné. Mnohé z nich je pred začatím projektu ťažké predpokladať. Veľký projekt je preto často finančne zabezpečovaný z rozpočtu. Aj tento síce vychádza z identifikovania zdrojov nákladov, no postačuje identifikovať len tie hlavné. Myslím si, že detailne sa zaoberať zdrojmi nákladov má význam hlavne pre stredne veľké projekty. Tu nevystupuje až toľko zdrojov nákladov a je predpoklad, že riešiteľ menšieho projektu operuje s menším zdrojom peňazí, a preto chce mať presnejšiu predstavu o celkovej cene. Pre malé projekty je často jediným zdrojom nákladov cena práce. Najjednoduchšie ju môžme získať vynásobením počtu odhadovaných hodín (získaných z odhadu úsilia) s jednotkovou cenou práce za jednu hodinu. Presnejšie výsledky získame, ak zoberieme do úvahy, že cena za prácu je rozdielna pre jednotlivých členov tímu projektu (napr. technik, projektový manažér, dokumentarista). Aj pri tomto odhadovaní nám môžu pomôcť údaje z minulosti.

## Presnosť odhadov

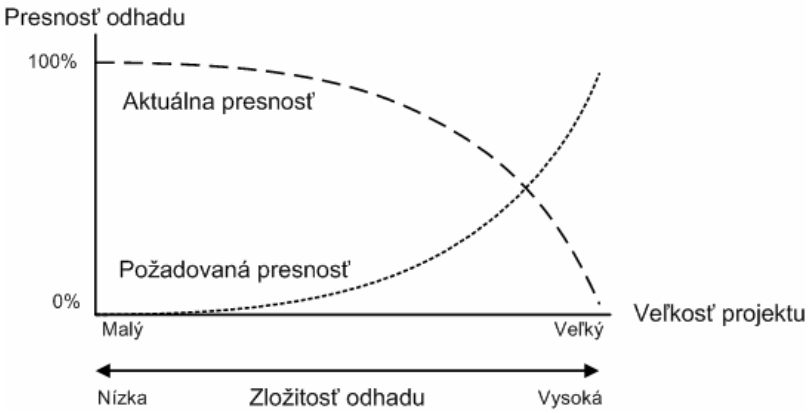
Manažér projektu teda po celom procese odhadovania má k dispozícii dve čísla, ktoré ho zaujímajú. Vie teda kedy bude produkt pripravený a koľko to bude stáť, no musí mať na zreteli že tieto čísla sú odhadnuté a podliehajú určitej nepresnosti. Výsledkom odhadov môže byť rozsah hodnôt alebo len jedna finálne hodnota. To, že odhad je reprezentovaný ako jedna hodnota neznamená, že odhad je presný. Rozsah hodnôt podľa mňa prináša viac informácií, hoci to znamená, že práca pri odhadovaní bude o niečo zdĺhavejšia. Pri odhadovaní intervalu budeme odhadovať namiesto jednej hodnoty dve, hornú hranicu a dolnú hranicu odhadu.

Definovanie odhadu ako rozsahu hodnôt má podľa [3] aj ďalšie výhody. Ak máme stanovený interval hodnôt, ktoré môže nadobúdať odhadovaná veličina, môžme tento interval počas práce na projekte zužovať a tým spresniť naše odhady. Počas projektu robíme ďalšie odhady, ktoré spresnia odhady vykonané pred začatím projektu (pozri **Obr. 1**).




**Obr. 1.** Odhadovanie počas priebehu projektu.

Podľa [2] je presnosť odhadu závislá od veľkosti projektu (pozri Obr. 2). Základný problém s odhadovaním je v tom, že malé projekty vieme odhadnúť veľmi ľahko, ale vyžadovaná presnosť nie je až taká dôležitá. Naopak veľké projekty je ťažké odhadnúť, ale vyžadovaná presnosť je veľmi dôležitá.




**Obr. 2.** Presnosť odhadovania v závislosti od veľkosti projektu.

## Vyhodnotenie odhadov

Aj keď sme poskytli manažérovi požadované odhady o čase a cene projektu, naša práca nekončí. Manažér projektu už na základe našich hodnôt rozhodol, či sa bude projekt realizovať, ale my sme počas procesu odhadovania získali okrem týchto dvoch hodnôt aj ďalšie dôležité dáta. Často sa stáva, že odhad je vytvorený, ale nevieme ho dostatočne efektívne využiť. Práve naopak, až teraz začne skutočná práca. Musíme nájsť ideálnu kombináciu funkcionality, rozvrhu, ceny a členov tímu tak, aby boli akceptované manažmentom projektu a zákazníkom. Je dôležité poznať vzťahy medzi týmito premennými a vplyv týchto vzťahov na projekt.

Ak predlžíme udalosti v kalendári, môžme znížiť celkové náklady a využiť menší tím. Niekedy stačí skrátiť kalendár o pár týždňov a získame tieto výhody. Obvykle manažment a zákazník nie sú ochotní dlho čakať, takže musíme vedieť aké oneskorenie bude ešte akceptované. Musíme však opäť odhadnúť, aký to bude mať dopad na projekt.

Kalendár je možné skrátiť tromi spôsobmi. Môžme redukovať funkcionalitu (znížime požadované úsilie), zvýšime počet členov tímu (iba ak je možné pracovať na úlohách paralelne) alebo ponecháme veľkosť tímu nezmenenú a budeme vyžadovať, aby pracovali nadčas. Ak nie je možné redukovať funkcionalitu, zameriame sa na optimalizovanie tímu. Je potrebné pamätať na pravidlo: "Pridanie inžinierov do oneskoreného projektu ho oneskorí ešte viac" [1]. V softvérovom projekte to znamená, že aj keď zvýšime počet členov tímu, zvýši sa zároveň aj množstvo práce, pretože sa zvýši rozsah komunikácie a manažovania tímu. Ak budeme vyžadovať od členov tímu, aby pracovali nadčas nemôžme očakávať, že to priamo úmerne skráti kalendár. Produktivita tímu síce v krátkom časovom úseku bude rásť, ale neskôr začne klesať, pretože ľudia budú unavení a budú robiť viac chýb.

Kombináciou zdrojov môžme takto meniť rozsah kalendára, ale musíme poznať hranice. Rozvrh kalendára nám musí dať možnosť implementovať všetky požadované funkcionality a otestovať ich, aby výstupný produkt dosahoval požadovanú kvalitu. Nemôžme prekročiť túto hranicu.

## Záver

Odhadovanie nie je jednoduchý proces a v projekte akejkoľvek veľkosti je potrebné mu venovať dostatočnú pozornosť. Výstupom odhadovania nie sú len hodnoty o dátume ukončenia projektu a cene projektu, ale aj ďalšie dáta, ktoré nám umožňujú využiť dostupné prostriedky v projekte čo najefektívnejšie. Kroky v procese odhadovania na seba nadväzujú, preto je potrebné dodržať ich správnu postupnosť.

## Použitá literatúra

1. Bieliková, M.: *Softvérové inžinierstvo – Princípy a manažment*. Vydavateľstvo STU, Bratislava, 2000.
2. Longstreet, D.: *Estimating Software Development*. http://www.softwaremetrics.com/Articles/estimating.htm, 2004. (2005)
3. Peters, K.: *Software Project Estimation*. http://www.software-engineer.org, 2000. (2005)

## Annotation

*Estimating software projects*

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without determining a reliable estimation. In general, software industry doesn't equally advert to estimating and doesn't use estimates appropriately. Incorrect estimating leads to incorrect use of resources, which results in missed deadlines and to waste of financial resources. Project management isn't often aware of that estimating isn't simple process, which has to define final value with one hundred percent accuracy. Estimating is always loaded by finite aberration and is specifying during project development. I attempt to finger in this article, which is the right progress of estimating, why is important and how to do estimation of different sized projects.

# Sledovanie postupu softvérového projektu

JÁN PORUBSKÝ

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
porubsky05@student.fiit.stuba.sk

**Abstrakt.** Softvérový projekt a jeho riadenie je dosť zložitý proces, pri ktorom sú dôležité viaceré postupy a metódy. Esej má za cieľ poukázať práve na jednu z týchto častí, na sledovanie postupu v softvérových projektoch. Snaží sa zdôvodniť potrebu sledovania projektu z hľadiska projektového manažéra a načrtnúť problémy spojené so slabým prehľadom o prebiehajúcom projekte. Esej postupne prechádza vysvetlením pojmu sledovania postupu projektu, manažérskym pohľadom na projekt a problémami, ktoré treba z tohto pohľadu riešiť.

## Prečo vlastne sledovať ?

Akýkoľvek projekt a samozrejme aj softvérový projekt je z hľadiska manažmentu značne náročná aktivita. Existujú väčšie aj menšie projekty a takisto z určitého pohľadu náročnejšie a menej náročné projekty. Či sa jedná o menší alebo väčší projekt, vždy je dôležité, aby bol pod kontrolou niekoho. Vždy stojí niekto za projektom, kto ho riadi. Táto úloha pripadá projektovému manažérovi a sledovanie postupu projektu je v jeho záujme a samozrejme aj v záujme úspešnosti celého projektu. K otázke potreby sledovať postup v softvérovom projekte by som chcel uviesť príklad nesprávnej komunikácie tímu a projektového manažéra.[1] *Ahoj Bill ako, ako ďaleko si s tým podsystémom? Ide to super , už je na 90% hotový. Na 90% ? Nehovoril si už minulý týždeň, že je na 90% hotový? Áno jasné, ale teraz je naozaj na 90% hotový.* Takýchto situácií môže byť veľké množstvo a myslím si, že iba veľmi ťažko projekt, v ktorom sa projektový manažér z času na čas zastaví a medzi rečou opýta „ako to ide“ uzrie úspešný koniec. Zaujímavý dôvod alebo odôvodnenie sledovania projektu uvádza aj [3], ktorý hovorí, že nemôžete niečo kontrolovať, bez toho aby ste to sledovali.

## Kto sleduje projekt ?

V úvode bolo z časti naznačené kto riadi projekt a dá sa povedať dozerá na všetko, čo sa v projekte deje. Predpokladom každého softvérového projektu je jeho úspešný koniec a samozrejme aj nejaký zisk z projektu. Inak by sa vývojom softvéru asi nikto

*Manažment v softvérovom inžinierstve, december 2005, s. 51-56.*

nezaoberal. Kto je vlastne ten, pre ktorého je sledovanie postupu v projekte také dôležité? Aký je rozdiel medzi ním a napríklad programátorom, ktorý pracuje na projekte? Rozdiel medzi týmito dvomi pozíciami nie je zaujímavý z pohľadu titulu pred menom alebo za menom, alebo sumy ktorú si nájde na konci mesiaca na účte. Dôležitý rozdiel je v tom akým spôsobom vidia projekt a jeho riadenie. Programátor vidí svoje technické problémy, vidí akoby jednotlivé časti reálne implementoval. Chýba mu celkový pohľad na projekt. Jeho nezaujíma rozdelenie jednotlivých zdrojov pre projekt, je mu v podstate jedno, že zákazník nevie špecifikovať čo vlastne chce a že to chce čo možno najrýchlejšie. On väčšinou dostane svoju úlohu, poprípade aj nejaký postup ako ju má realizovať a pracuje na úlohe.

Predstavme si že máme stredne veľký projekt, v ktorom máme iba programátorov. Pre jednoduchšiu predstavu všetky požiadavky zákazníka sú už zistené a všetkým viac alebo menej jasné. Celý tím sa skladá iba z programátorov, ktorí sú síce výbornými odborníkmi vo svojom zameraní ale nemajú potrebný nadhľad. Aby boli viac ako programátormi, chýbajú im schopnosti pre vedenie projektu. Čiže náš fiktívny projekt sa rozbehne, samozrejme neexistuje žiadny plán čo má kto robiť ani ako dlho to bude trvať. Proste jednotlivé úlohy si zoberie ten komu sa páčia. Neexistuje odhad koľko bude projekt stáť. Komunikácia so zákazníkom a medzi jednotlivými členmi tímu je iba náhodná ak vôbec nejaká je. Výsledkom takéhoto projektu, ak by mal aj nejaký výsledok, by bolo možno niečo zlepené, čo by v lepšom prípade aj fungovalo. Zákazník by sa dozvedel termín dokončenia až vtedy, keď by mu výsledný projekt niekto priniesol a nejakým nezrozumiteľným jazykom vysvetľoval čo s tým „niečím“ má robiť. Toto by bol ten lepší prípad. S väčšou pravdepodobnosťou by takýto projekt skôr skončil, ako by začal.

Čiže projekt, na to aby bol projektom potrebuje softvérového inžiniera, spolu s jeho schopnosťami a skúsenosťami na to, aby bol projektom a nie iba nejakým zhlukom pokusov vytvoriť niečo. Takže máme softvérového inžiniera v osobe projektového manažéra, ktorý má na starosti projekt. Plán projektu je vytvorený, jednotlivé úlohy sú rozdelené a sú v procese vytvárania. Zdalo by sa že týmto sa práca projektového manažéra končí a stačí počkať iba na výsledný produkt projektu. Realita je ale o dosť iná. Je veľa faktorov ktoré vplývajú na projekt a môžu ho zmeniť z perfektne fungujúceho na katastrofálny chaos v priebehu pár hodín. Samozrejme to je už až extrémny prípad, ktorý ale nie je nemožný. Skôr pravdepodobnejšie možnosti sú rôzne malé problémy a omeškanie, ktoré sú samé o sebe možno až zanedbateľné detaily, ale spolu vytvárajú nemalé komplikácie v projekte. Aj keď projektový manažér je súčasťou projektu a podieľa sa na ňom v značnej miere, nemôže byť pri všetkých častiach a fázach projektu. Taktiež môže súčasne pracovať na viacerých rozdielnych projektoch naraz.

Čiže týmto sa dostávame k potrebe softvérového inžiniera sledovať určitými spôsobmi postup v projekte aby si udržal o ňom prehľad a bol schopný reagovať na vzniknuté situácie a prijímať potrebné opatrenia.

## Čo je potrebné sledovať ?

Existujú základné požiadavky, aby vôbec sledovanie projektu mohlo prebiehať, alebo lepšie povedané, aby bolo čo sledovať. Medzi tieto požiadavky patrí plán projektu, určenie nákladov na projekt, rozdelenie a priradenie jednotlivých úloh. Ak ale teda máme v projekte existujúci plán, ktorý vyzerá výborne, na čo ho je potrebné ešte pravidelne sledovať? Odpoveď na túto otázku je dobre zachytená v [1]. *Mať plán softvérového projektu je jedna vec, ale dodržať jeho postup je vec druhá.* Je dosť pravdepodobné, že aj keď bude plán projektu vypracovaný do úplných podrobností a na vysokej úrovni, vzniknú odchýlky od plánu, poprípade bude potrebné ho v určitom smere prerobiť alebo mierne pozmeniť. Projektový manažér nikdy nemôže dopredu vedieť, čo sa počas projektu zmení a aké možné prípady nastanú. Aj keď sa dobrý projektový manažér často označuje ako jasnovidec alebo vizionár, lebo by mal byť schopný počas projektu vidieť aj dopredu a z časti situáciu predvídať, plán bude s veľkou pravdepodobnosťou aj tak mierne meniť a upravovať.

Z hľadiska sledovania projektu je dôležité zistenie aktuálneho stavu, v ktorom sa projekt nachádza. Projektovému manažérovi nemôže stačiť zastaviť nejakého programátora na chodbe a opýtať sa ho, ako je na tom s projektom a koniec. Potom ľahko môže nastať situácia uvádzaná v úvode, že každý bude hotový už na 90 percent tak isto ako pred mesiacom. Projektový manažér musí zistiť reálne čo je na projekte hotové, podľa jednotlivých úloh a stavu v akom sú. Dôležité je aj porovnanie týchto údajov s tým, čo bolo plánované a aké boli očakávania stavu projektu podľa plánu. Dá sa povedať, že projektový manažér zoberie plán a stav projektu podľa jednotlivých úloh a porovná ich. Informácie o projekte by mal získať na dvoch úrovniach. Na úrovni jednotlivých úloh a následne potom na úrovni celého projektu.

Myslím si, že zistenie aktuálneho stavu projektu je kľúčová časť procesu sledovania projektu. Projektový manažér zistí v akom stave sa projekt nachádza oproti pôvodnému plánu. Stav projektu je ale potrebné sledovať z viacerých pohľadov alebo podľa viacerých kritérií. Ako prvé kritérium by mohol byť práve čas, ktorý bol spotrebovaný od začiatku projektu. Čas je často veľmi často dosť veľkým obmedzením pre projekt a všetkých zúčastnených na projekte. Pri sledovaní projektu je nutné zistiť ako dlho jednotlivé fázy trvali a tiež koľko času zostáva do konca projektu. Určite by nebolo dobré, ak by projektový manažér zistil po uplynutí doby určenej pre projekt, že je hotová sotva tretina vecí ktoré sa mali urobiť.

Ďalším nie menej významným objektom na sledovanie sú náklady na projekt a na jednotlivé úlohy. Náklady na projekt vychádzajú väčšinou z odhadu na začiatku projektu. Tento odhad závisí na skúsenostiach manažéra a podľa jeho schopností je viac alebo menej presný. Povedal by som, že náklady na projekt veľmi významne vplývajú na úspešnosť projektu. Čiže sledovanie nákladov má svoje odôvodnenie a projekt, v ktorom sa počas trvania spotrebuje dvojnásobok ako boli pôvodné náklady, bude možno úspešne ukončený, ale je otázne či bude spokojný zákazník a či to nebude posledný projekt pre takéhoto projektového manažéra.[4]

Takže náš projektový manažér už zistil nejaké informácie o stave projektu. Je mu jasné koľko času trvala daná fáza projektu a čo sa počas nej vyprodukovalo. Ďalej vie, aké boli náklady na doterajšiu činnosť na projekte. Svoj záujem by mal upriamiť aj na

jednotlivé úlohy a na to komu sú priradené. Jedna vec je, že na začiatku boli rozdelené jednotlivé pozície alebo dá sa povedať roly v jednotlivých úlohách tým ľuďom, ktorých mal manažér k dispozícii. Počas projektu mohlo prísť k nejakým zmenám alebo práve sledovanie projektu odhalilo, že na nejakú úlohu treba nasadiť viacerých ľudí alebo naopak, pracuje na nej veľký počet ľudí. Proste sledovanie by malo slúžiť aj na vyťaženie čo najväčšieho výkonu z daného tímu, ktorý pracuje na projekte.

V podstate pri otázke čo sledovať sa dá odpovedať rôzne veľkou množinou alternatív. Je práve na manažérovi aby si zvolil pre neho potrebnú a zaujímavú skupinu vecí a sledoval ich do takej miery ako potrebuje. Myslím si, že projekt by sa v žiadnom prípade nemal zmeniť na preteky v sledovaní. Aj tu platí, že nakoniec všetkého veľa škodí.

## Ako sledovať ?

Doteraz som hovoril o tom ako a kto sa zúčastňuje na sledovaní, alebo kto je ten, pre koho je sledovanie najdôležitejšie a čo je vlastne zaujímavé sledovať. Ďalej je sa vynára otázka ako sledovať ?

Ak by sme si predstavili situáciu, v ktorej prebieha nejaký projekt. Úlohy sú rozdelené a tí, ktorým boli predelené na nich aj pracujú. Projektový manažér sa rozhodne sledovať projekt, aby zistil v akom stave sa nachádza. Ešte pred začatím projektu nainštaloval do každej kancelárie kameru a vo svojej kancelárii sleduje každý krok, ktorí jeho zamestnanci spravia. Možno by takýto postup na niečo bol, ale pochybujem, že by sa to ľuďom na projekte páčilo a boli by schopní takto pracovať. Čiže pre náš, dúfajme aj úspešný projekt budú zaujímavejšie iné a lepšie metódy sledovania.

### Metrika

Meranie je významná činnosť pre ľudí od dávnych čias až dodnes. Človek si za tento čas zvykol na to, že sa má lepšiu predstavu o veciach, ak je schopný vyjadriť ich vlastnosti v číslach. Takisto ako je meranie dôležité napríklad pri stavbe budovy je dôležité aj v softvérovom inžinierstve. Softvérová metrika sú numerické dáta vzťahujúce sa k projektu.[2]

Metrika v softvérovom projekte je veľmi dobrý nástroj pre lepšie porozumenie projektu, ale voľba a implementácia správnej metriky je takisto veľká výzva.[2] Meranie v softvérovom projekte vlastne začína už pri tvorbe plánu, pri ktorom sa tiež jednotlivé úseky vyjadrujú v číslach. Taktiež ako pri celom procese sledovanie aj pri metrike existuje veľký výber možných metrík. Meranie nemusí byť použité iba pri sledovaní postupu v projekte. Využitie postupov meranie sa dá nájsť dá sa povedať skoro vo všetkých fázach projektu.

Zaujímavý je názor uvedený v [2], že vo viacerých oblastiach riadime veci lepšie prostredníctvom čísel. Čiže ak projektový manažér má vo svojej pozícii zistiť aktuálny stav projektu, má ako jeden veľmi silný nástroj práve meranie. Samozrejme je na ňom pre akú metriku sa rozhodne, či to bude rátanie počtu riadkov zdrojového kódu alebo

niečo iné, myslím že meranie je veľmi rýchla cesta na zistenie stavu projektu. Nechcem sa venovať jednotlivým spôsobom a typom nerania, hlavný cieľ je zdôrazniť to, že meranie zastáva pri sledovaní projektu významnú úlohu.

**Nástroje na sledovanie**

Existuje veľké množstvo projektov. Niektoré sú jednoduchšie a tým pádom je aj ich manažment menej komplikovaný. Sledovanie zložitejších projektov si vyžaduje viac úsilia v každej fáze projektu. Myslím si, ž fáza sledovania postupu projektu nie je výnimkou z tohto pravidla.

Ak má projektový manažér na starosti projekt väčších rozmerov, má k dispozícii celkom slušné množstvo nástrojov, podporujúcich sledovanie údajov o postupe projektu. Používanie týchto nástrojov určite zlepšuje prehľad v celom projekte a umožňuje jednotlivé výsledky a hodnoty prezentovať v zaujímavejšej a možno krajšej podobe. Možno je celkom zaujímavé a vhodné, v určitej fáze alebo období prezentovať zistený postup v projekte aj zákazníkovi. Samozrejme v prípade, že postup na projekte je v takom stave, akým by sa chcel projektový manažér pochváliť. Aj keď na druhej strane, ak vznikne problém v projekte, z ktorého napríklad vyplýva značné predĺženie projektu, takáto prezentácia môže tiež pomôcť v komunikácii so zákazníkom.

Nástrojov, ktoré pomáhajú pri sledovaní projektu je veľké množstvo a dá sa vyberať medzi rôznymi verziami, spôsobmi zobrazenia a funkciami. Jednotlivým nástrojov sa venovať nebudem, podľa mňa je dôležité vedieť, že takéto nástroje existujú a sprehľadňujú prácu na projekte.

## Čo potom ?

Čiže projektový manažér využil pre neho zaujímavé metódy sledovania postupu projektu. Zistil aký je stav projektu a porovnal ho s tým čo bolo naplánované. Čo ale s týmito údajmi ?

Ako bolo spomenuté už v predošlých častiach, aj keď je plán projektu akokoľvek dobre vypracovaný, nepôjde všetko podľa neho. Práve informácie o stave projektu sú dôležité pre zistenie ako je na tom projekt v porovnaní s plánom. Projektový manažér vyhodnotí to čo bolo naplánované a to čo sa naozaj reálne spravilo. Pri tomto kroku môžu nastať rôzne situácie. Napríklad ak sa súčasný stav projekt príliš odkláňa od plánu je potrebné zasiahnuť a prijať potrebné rozhodnutia. Rozhodnutia o zmenách sú na projektovom manažérovi, pričom plán nie je vec ktorá je nemenná. Myslím si, že plán je potrebné pravidelne meniť a prerábať podľa aktuálnej potreby a výsledkov sledovania projektu.

Takže ako výsledok úsilia sledovania projektu, má projektový manažér informácie o tom, koľko práce sa doteraz urobilo, to či je potrebné meniť plán projektu ale robiť ďalšie zásahy do organizačnej štruktúry projektu. Okrem toho vie, aj keď určite nie úplne presne, to koľko práce je ešte potrebné v projekte spraviť a koľko

zdrojov bolo v projekte doteraz spotrebovaných. Takže údaje o postupe v projekte sú podľa mňa dôležitým prvkom pre ďalší postup v projekte a základ pre rozhodovanie.

## Záver

Myslím si, že projekty či väčšie alebo menšie, sú z hľadiska manažmentu náročnou výzvou. Pre projekt je dôležitý odhad času trvania, nákladov a rozdelenie jednotlivých úloh. Práve manažér je ten, ktorý to má všetko odhadnúť a riadiť a samozrejme aj byť za to zodpovedný. Manažér pri projekte využíva svoje schopnosti získané praxou, ale do určitej miery aj svoj cit alebo schopnosť predvídať určité udalosti.

Sledovanie postupu projektu je pre jeho úspešné ukončenie veľmi dôležité. Ak si predstavíme, že projektový manažér má na starosti celý projekt a samozrejme nemusí to byť iba jeden projekt na ktorom pracuje, potrebuje niečo podľa čoho je schopný sa rozhodovať v tomto projekte.

## Použitá literatúra

1. Karl E. Wiegers: *Project management best practices*. Dostupné na http://www.processimpact.com/PMBP/ (December 2005)
2. Michael C. Mah, Lawrence H. Putnam: *Software by the numbers*. Dostupné na http://www.qsm.com/arialview.html (December 2005)
3. Ryan Lowe: *Software project monitoring can facilitate creativity*. Dostupné na http://www.ryanlowe.ca/blog/archives/2004_12.php (December 2005)
4. *Software Project Tracking and Oversight process*. Dostupné nahttp://sepo.spawar.navy.mil/SPTO.doc (December 2005)

## Annotation

*Tracking progress in software project.*

Software project and it's management is quite difficult process, in which are more methods very important. The main aim of essay is to show one of these parts, which is tracking progress in software projects. This essay try to show how is needful tracking progress in project by project's manager and show problems connect with remote perspective about project. Essay should to explain expression of tracking in project, manager's view of project and problems which must be solved.

# Agilné metódy vývoja softvéru a rozsah projektu

TOMÁŠ KLEMPA

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
klempa05@student.fiit.stuba.sk

**Abstrakt.** Tradičné metódy vývoja softvéru už prestávajú byť postačujúce, v súvislosti s požiadavkou na rýchly vývoj softvéru a jeho časté zmeny. Zákazníci požadujú dodávku softvéru v čo možno najkratšom čase a v čo možno najvyššej kvalite (a primeranej cene). Čas a kvalita sú premenné projektu, ktoré sú ovplyvnené rozsahom projektu. Rozsah projektu je spočiatku pevne stanovený, ale po istom čase sa začne sa začne meniť, čo môže ohroziť úspech projektu. So zmenou rozsahu však treba vopred počítať. Podľa zdrojov použitých v tejto eseji vyplýva, že riadenie rozsahu projektu možno efektívnejšie docieliť s použitím agilného vývoja. Esej sa zaoberá otázkou čo nám môžu agilné metodológie ponúknuť, definuje rozsah a s nim súvisiace základné premenné projektu, naznačuje spôsob ako možno efektívnejšie riadiť rozsah projektu, vysvetľuje vplyv neistoty a zložitosti v projekte. V závere je spomenutých niekoľko tipov, ktoré by mohli napomôcť pri agilnom vývoji v súvislosti s rozsahom projektu.

## Úvod

Známy fakt, že softvér sa mení vplyvom okolia je skutočne pravdivý. Zákazník chce dostať za svoje peniaze vysokú kvalitu, a keď ju aj dostane po istom čase (alebo aj ihneď) by chcel mať vo svojom softvéri toto a toto, napr. kvôli potrebám spoločnosti. Všeobecne povedané ide o vylepšenie, alebo zmenu softvéru. Zákazník chce vymeniť zastaraný systém, ktorý už nevyhovuje jeho potrebám. Softvér sa musí prispôsobiť keďže zákazník žiada zmeny v softvéri.

Nasleduje identifikácia nových čŕt, ktoré by mal „vylepšený“ systém obsahovať. Dátumy vývoja boli stanovené na základe najlepších odhadov času potrebného pre vývoj týchto čŕt. Nasleduje podrobný návrh týchto čŕt, čo je veľmi dôležité. V prípade, že bude nová črta implementovaná nekorektne (rozmaznaný zákazník si to predstavoval ináč), spôsobí to oneskorenie odovzdania systému. Celý projekt môže byť vážne ohrozený. Kvalita produktu je veľmi dôležitá a pre dodávateľa snáď nejestvuje nič horšie ako strata dôvery zákazníka.

*Manažment v softvérovom inžinierstve, december 2005, s. 57-65.*

Oneskorený projekt bude dodaný načas, ale vývojový tím bude pracovať nadčasy, čo sa odrazí na ich produktivite. Reťazová reakcia určite platí aj v softvérom inžinierstve a teda zníženou produktivitou utrpí aj kvalita projektu, v dôsledku čoho môže nastať prípad, kedy zákazník „objaví“ chyby zmeneného systému, ktorý tak túžobne očakával. Pravdepodobne ešte horší prípad môže nastať, keď sa dodanie systému oneskorí (aj o niekoľko mesiacov!). To už zákazník pení a v jeho očiach dodávateľ stráca kredit, čo môže vyústiť do stavu, kedy sa zákazník v budúcnosti bude rozhodovať pre systém konkurenčného dodávateľa.

Iba prednedávnom sa objavili na „scéne“ agilné metódy vývoja softvéru. Nie sú však všeliekom na neduhy softvérového inžinierstva a ani sa o to (našťastie) nesnažia. Poďme sa pozrieť bližšie na agilné metódy a čím môžu byť užitočné pre nás.

## Čo nám môžu ponúknuť agilné metodológie

Dovolím si stručne ozrejmiť pojem agilné metodológie, aby v ďalšom texte nedošlo k nejednoznačnosti. Agilné metodológie označuje skupinu metodík, ktoré vychádzajú z poznania, že jedinou cestou ako overiť správnosť navrhnutého systému, je čo najrýchlejšie ho vyvinúť, predložiť zákazníkovi a na základe spätnej väzby upravovať [4].

Agilné metodológie sľubujú zlepšenie výkonu projektu. Neznie to príliš nadnesene? Sľuby v softvérovom inžinierstve vyznievajú príliš nedôverčivo, ale skúsime sa pozrieť bližšie.

Praktiky ako *testom riadený vývoj*, *jednoduchý návrh* a *refaktoring* môžu zvýšiť kvalitu kódu.

Krátke iterácie nám ponúkajú viac informácii o tom, ako sme na tom s projektom a poskytujú rýchlu spätnú väzbu od zákazníka.

*Scrum* praktiky definujú denné stretnutia a pravidelné demonštrácie na konci mesiaca. Zapájajú aj zákazníka minimálne raz do mesiaca.

*Informačné radiátory* (radiators) predstavujú udržiavanie informácií o *interakčnom (interactive) návrhu*, objektových modelov a iteračných plánov. Informácie sú umiestnené fyzicky na stene kde umožňujú kontinuálne informovať tím.

Niekoľko atribútov *extrémneho programovania (XP)*: programovanie v pároch (zvýšenie kvality kódu), krátke iterácie a neustále testovanie.

Pre detailnejšie popisy jednotlivých praktík odporúčam čitateľovi naštudovať si príslušnú literatúru, napr. aj [1].

### Druhá strana mince alebo štatistiky neklamú

Zaujala ma štatistika [3] o znalostiach projektových manažérov zo spoločností, ktoré používajú tradičné metodiky vývoja (plan-based). Oslovených bolo 20 manažérov (10 tradičných a 10 agilných spoločností), pričom ich priemerný vek je 40 rokov. 14 spoločností sídlia v Taliansku, 5 v USA a 1 v Švajčiarsku. Štatistika hodnotila znalosti manažérov o agilných metódach, ich výhodách a nevýhodách.

Približne 90% manažérov vedia o agilných metódach, hoci ich nepoužívajú. Hlavnou príčinou pre neprijatie agilných metód sú povrchné znalosti o nich, odpor v spoločnosti a zákazníkov a veľké geograficky oddelené tímy. Odpor zo strany zákazníkov sa mi pozdáva nemiestny – od kedy určujú zákazníci aké metódy má dodávateľ používať? Zrejme netušia, že agilné metódy kladú dôraz na vývoj softvéru spolu so zákazníkom.

Ďalšia otázka sa týkala znalostí o XP a metóde Scrum. Všetky spoločnosti, ktoré používajú tradičné aj agilné metódy vývoja softvéru poznajú XP, ale iba 60% agilných spoločností a žiadne z „tradičných“ spoločností poznajú metódu Scrum.

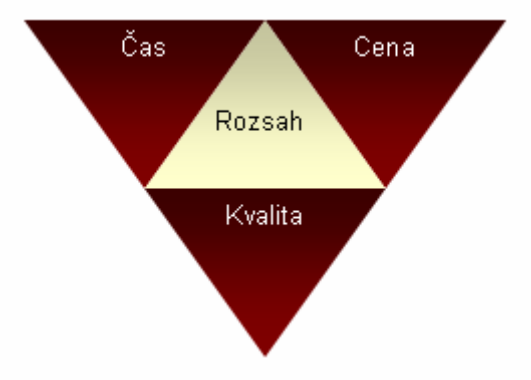
Čísla pôsobia až šokujúco, autori v článku tvrdia, že výsledky tejto štatistiky nereprezentujú celkovú populáciu v softvérových spoločnostiach.

## Štyri premenné

Vychádzajme z faktu, že existujú štyri základné premenné vývoja softvéru (podľa [1]):

1. cena,
2. kvalita,
3. čas a
4. rozsah.

Čas a cena sú v podstate výstupom rozsahu. Ak máme definovaný rozsah, môžeme vypočítať čas (ako dlho nám to bude trvať) a cenu (koľko nás to bude stáť). Nesprávne definovaný rozsah negatívne ovplyvní čas a cenu.




**Obr. 1.** Štyri premenné projektu.

Premenné je vhodné v skratke opísať a vysvetliť ako vzájomne pôsobia.

### Cena

Túto premennú možno ovplyvniť zmenou ľubovoľnej inej premennej. Veľký vplyv na cenu majú ľudia participujúci na projekte. Je zrejmé, že pridaním dvojnásobku počtu

ľudí sa vývoj projektu nezrýchli, naopak, je pravdepodobné, že sa spomalí vzhľadom na problémy komunikácie v rámci tímu, keďže dvojnásobná bude komunikácia v tíme.

V poriadku, pridajme teda do projektu niekoľko ľudí (menej ako dvojnásobok) a uvidíme čo sa bude diať. Pri stavbe domu by sa práca pravdepodobne urýchlila, ale to pri vývoji softvéru v tejto situácii opäť neplatí. Efekt by sa dostavil až po dlhšom čase. Podstata tkvie v tom, že novému členu tímu chvíľu trvá, než spozná zabehnutý systém vývojového tímu. Hoci urobí určitú časť práce za iných členov tímu, členovia tímu mu musia isté veci vysvetliť a tým ich akoby okrádal o čas. Dovolím si citovať Fredericka P. Brooksa: „Pridaním ďalších ľudí do projektu, ktorý mešká, ho urobí ešte viac meškajúcim.“ [2].

### Kvalita

Podľa [1] možno rozlíšiť kvalitu na:

- externú – viditeľná zákazníkom (napr. nefunkčné požiadavky) a
- internú – určuje „vnútorné” kvality systému: dobrý návrh, testovanie atd. Vnútorná kvalita je veľmi dôležitá a má vplyv na rýchlosť vývoja projektu (chabá vnútorná kvalita má za následok spomalenie). Z tohto dôvodu napr. kladie XP dôraz na testovanie a refaktoring.

### Čas a rozsah

Sú premenné najvhodnejšie pre riadenie, ktoré sú spočiatku fixné, ale neskôr sa musia nevyhnutne meniť a narastať (napr. nové podrobnosti, ktoré nie sú na začiatku predvídateľné), čím môže utrpieť kvalita. A následkom toho, že čas a rozsah sa mení, vieme presne povedať, kde sme až na konci projektu.

Z tohto dôvodu sa javí ako výhoda projekt „ukončovať” v pravidelných intervaloch (rádovo v týždňoch) – iteráciách. Iterácie nám pomáhajú sledovať účinky meniaceho sa rozsahu projektu, ktoré môžu byť fatálne pri tradičných metódach vývoja softvéru (pri použití vodopádového modelu).

## Outsourcing a plánovanie v XP

Outsourcing je v súčasnosti veľmi frekventovaný pojem. Akým spôsobom však ovplyvňuje rozsah projektu? Odpoveď nájdeme v outsourcingovej zmluve. Podľa [1] obvyklá outsourcingová zmluva stanovuje tri zo štyroch základných premenných: rozsah, čas a cenu. Tradičné biznis vzťahy potrebujú miernu úpravu, pokiaľ chceme plánovať a vykonať projekt s XP.

Termín “premenná” nie je použitý náhodou. Ak sa objaví nečakaná situácia, jedna z premenných sa zmení (napr. čas). Od kvality, ktorá je defacto najťažšie zmerateľná premenná, nečakané situácie majú tendenciu ovplyvniť práve ju. Prakticky to znamená menej testovania, menej návrhu, a menej komunikácie v rámci tímu, alebo so zákazníkom. Akonáhle máme neznáme množstvo práce, rozsahu, času a ceny, úspech projektu je vážne ohrozený.

Autori [1] vidia základný problém v pevne stanovenej zmluve a to zmysle, že vo svojich záujmoch idú dodávateľ softvéru a zákazník navzájom proti sebe. Zákazník chce za svoje peniaze čo možno najväčší rozsah, kým dodávateľ chce urobiť čo možno najmenej práce. Tento fakt však nemusí výrazne ovplyvniť projekt, pokiaľ ide vývoj podľa plánu. Situáciu možno opísať nasledovným príkladom: zákazník podpíše zmluvu s dodávateľom a za svoje veľké peniaze, ktoré zaplatil trvá na tom, aby mu dodávateľ presne oznámil čo dostane. Má na to samozrejme právo, ale kde je potom problém? Plánovanie je nie je predvídanie budúcnosti [1]. Táto veta, myslím, hovorí za všetko. Zákazník môže sledovať vývoj v pravidelných intervaloch, napr. každé dva týždne.

## Dodanie softvéru v čas a vo vysokej kvalite?

Jeff Patton v [6] sa zaoberá, akým spôsobom možno docieliť efektívnejšie riadenie rozsahu projektu. Použil agilný vývoj a interakčný (interaction) návrh.

Interakčný návrh predstavuje množinu praktík, ktorých prínosom je primerané správanie systému. Návrh možno rozdeliť na dve fázy. Prvá fáza sa zameriava na ľudí a skúma čo je ich cieľom. V ďalšej fáze, ktorá nadväzuje na získané ciele, vytvoríme najmenšiu a najjednoduchšiu množinu úloh, ktoré umožňujú ľudom používať ich softvér. Zámerom je teda splniť tieto ciele. Interakčný návrh je svojou koncepciou výrazne odlišný od tradičného návrhu.

Prínos autorovho príspevku vidím v skutočnosti, že agilné metodológie môžu za určitých podmienok zlepšiť riadenie rozsahu projektu. Pokladal som za dôležité, v skratke interpretovať autorovu ideu. Odporúčania vyplývajúce z tohto článku sú napísané v kapitole Recept na spokojný život alebo rady pri agilnom vývoji.

Ak čitateľa táto myšlienka zaujala, odporúčam prečítať si článok [6].

## Zložitosť a neistota projektu

Zaujal ma článok [5], v ktorom autor kategorizuje projekty do štyroch skupín v súlade s ich zložitosťou a neistotou.

Autor na základe projektov riešených v minulosti zistil, že existujú dva základné atribúty, ktoré ovplyvňujú typ použitého procesu: zložitosť a neistota.

Začal vyšetrovaním Crystal metódy Alistaira Cockburna a aj prístupu Barryho Boehma založeného na riziku (z dôvodu získania väčšieho množstva atribútov projektu), pre zmiešanie agilného a plánom riadeného (tradičného) vývoja softvéru. Vymenoval všetky kritické atribúty a zistil dva základné: zložitosť a neistotu.

V nasledovných dvoch podkapitolách opíšem základnú myšlienku zložitosti projektu a detailnejšie neistotu projektu, ktorá súvisí aj s flexibilitou rozsahu projektu.

### Zložitosť

Autorov tím vyvinul systém pre bodovanie zložitosti projektu na základe: veľkosti tímu, nevyhnutnosti úlohy, sídla tímu, vyzretosti tímu (výkonnosti tímu), rozdielov medzi znalosťami v oblasti a závislosti s ďalšími projektmi.

*Veľkosť tímu*

Veľkosť tímu ma majoritný vplyv na zložitosť projektu. Ako som uviedol v predchádzajúcom texte, pridaním dvojnásobného počtu ľudí do tímu, projekt dvojnásobne neurýchli. Často sa vedú diskusie o tom, či sú agilné metodológie vhodné pre veľký tím, to už je však otázka pre inú esej.

*Dôležitosť projektu*

Dôležitosť projektu má tiež majoritný vplyv na vývoj. Je zrejmé, že zložitosť projektu bude narastať spolu s jeho dôležitosťou.

*Sídlo tímu*

Sídlo tímu v jednej miestnosti umožňuje výbornú komunikáciu medzi členmi tímu. Naopak, tímy rozdelené vo väčších vzdialenostiach, alebo tímy sídliace vo viacerých časových pásmach zvyšujú zložitosť projektu.

*Výkonnosť tímu*

Veľký rozdiel medzi tímom zloženým zo skúsených vývojárov a tímom zloženého z úplných začiatočníkov.

*Rozdiely medzi znalosťami v oblasti*

S klesajúcimi znalosťami vývojárov v danej oblasti narastá zložitosť projektu.

*Závislosti*

Atribút určuje stupeň závislosti projektového tímu na ďalších skupinách alebo na iných projektoch v spoločnosti. Tesná spolupráca s ďalšími projektmi môže zvýšiť zložitosť projektu.

**Neistota**

Základné atribúty neistoty projektu (pozri obrázok č. 2) sú: neistota trhu, technická (odborná) neistota, dĺžka projektu a iné projektové závislosti na danom projekte a flexibilita rozsahu projektu.

*Neistota trhu*

Ak sú potreby trhu dobre známe, projekt nebude potrebné príliš riadiť a naopak. Riziko môže predstavovať prienik spoločnosti na nový trh.

*Technická neistota*

Projektové tímy vyvíjajúce nové produkty často využívajú najnovšie technológie, takže budú mať vyšší stupeň neistoty. Vyzreté produkty používajú overené technológie, a preto budú mať minimálnu technickú neistotu, hoci sa môžeme stretnúť so zvýšením technickej neistoty pri pridaní novej technológie do existujúceho produktu.

| Atribúty neistoty a ich bodové ohodnotenie | | | | | |
|---|---|---|---|---|---|
| Atribút | 1 | 3 | 5 | 7 | 10 |
| Neistota trhu | Známy produkt, pravdepodobne definovaný zmluvný záväzok | Očakávaná minimálne zmeny na cieľovom trhu | Počiatočný odhad cieľového trhu pravdepodobne k požiadavke riadenia | Podstatná neistota trhu | Nový, neznámy a neodskúšaný trh |
| Technická neistota | Zlepšenie existujúcej architektúry | Vieme ako to vytvoriť | Nie sme si celkom istý ako to vytvoriť | Zložitý prírastkový výskum | Nová technológia, nová architektúra |
| Dĺžka projektu | 1 - 4 týždne | 6 mesiacov | 12 mesiacov | 18 mesiacov | 24 mesiacov |
| Závislosti, flexibilita rozsahu | Dobre definované zmluvné záväzky | Rozsah nie je veľmi flexibilný | Rozsah má miernu flexibilitu | Rozsah je veľmi flexibilný | Veľmi vysoká závislosť |

**Obr. 2.** Atribúty neistoty a ich bodové ohodnotenie, zdroj: [5].

*Dĺžka projektu*

Čím bude dĺžka projektu vyššia, tým väčšia je šanca, že produkt ovplyvní technická alebo trhová neistota. Podľa definície rozsahu a obrázku č.1 je dĺžka projektu ovplyvnená rozsahom projektu. Navyše, ak sa dodanie produktu oneskorí, môže tým utrpieť kvalita.

*Závislosti a flexibilita rozsahu*

Vyšší stupeň závislosti nášho projektu na iných projektoch môže obmedziť úroveň riadenia projektu. To samozrejme súvisí s rozsahom projektu (vyššia flexibilita spôsobí vyššiu neistotu).

## Recept na spokojný život alebo rady pri agilnom vývoji

Autor [6] na základe vyriešených projektov, vytvoril zoznam, ktorý sa snaží nasledovať pri vývoji nového projektu. Zoznam ma zaujal a z tohto dôvodu pokladám za vhodné uviesť niekoľko bodov, ktoré by mohli pomôcť čitateľa nasmerovať pri agilnom vývoji softvéru. Obsahy jednotlivých bodov som opísal vlastnými slovami.

- *Ponechajte návrh všeobecný a rozsah jemný.* Autor odporúča identifikovať ľudí, ktorí budú používať systém, úlohy a priority. Je známe, že pri agilných metódach sa takmer zatracuje návrh a preto autor odporúča vyvarovať sa podrobnému návrhu obrazoviek, návrhu tabuliek databáz a pod. Tieto detaily môžu rozptyľovať pri detailnej identifikácii čŕt a ich priorít.
- *Zákazník nie je protivník.* Ak sa chce zákazník podieľať a pomôcť, neexistuje dôvod prečo ho odmietnuť!

- *Vytvorte plán spolupráce.* Vytvorenie plánu spolupráce (collaboration) so zameraním na detaily. Plán by mal obsahovať činnosti, ktoré budú riešené spolu so zákazníkom napr. spoločné stretnutia. Ďalší bod, ktorý potvrdzuje môj názor, že agilné metódy sú „ľudskejšie".
- *Fázy dodávky.* Autor odporúča rozdeliť projekt aspoň na dve fázy (v nezávislosti na veľkosti projektu). Prvú fázu dodávky by si mal zákazník možnosť vyskúšať. Úspech prvej fázy projektu zvýši dôveru zákazníka v projekte a tíme. Prvá fáza navyše umožňuje odhaliť, čo zákazník naozaj požaduje od produktu a čo môže byť v produkte vynechané. To sa odrazí vo finálnej dodávke.
- *Plánujte aj vynechanie čŕt.* Uvoľnenie každej fázy projektu by malo obsahovať aj niekoľko čŕt s nízkou prioritou. Konštrukcia softvéru by mala dovoliť jednoduché odstránenie, alebo znemožnenie nekompletných čŕt. Taktiež by mala byť dodávka softvéru možná.

## Záver

Ako z textu vyplýva, agilné metodológie skutočne môžu efektívnejšie „riadiť" rozsah projektu. Vhodnými praktikami možno predísť (alebo ich obmedziť) negatívnym následkom ako napr. oneskorenie dodania produktu, alebo pokles jeho kvality.

Počnúc od plánovania až po dodávku produktu je výhodná aktívna spolupráca so zákazníkom. Ako pozitíva by som zdôraznil hlavne iteračný vývoj a pravidelné stretnutia so zákazníkom spojené s prezentáciou už hotovej časti produktu.

Myslenie projektových manažérov neovplyvním, ale myslím si, že by nemali zatvárať oči pred novými metódami, ktoré by mali efektívnejšie pomáhať pri vývoji softvéru. Ich spoločným cieľom je predsa dodávka kvalitného produktu a dôvera zákazníka.

## Použitá literatúra

1. Beck, K., Fowler, M.: *Planning Extreme Programming First Edition*. Addison-Wesley, 2000.
2. Brooks, F. P.: *The Mythical Man-Month: Essays on Software Engineering Anniversary Edition*. Addison-Wesley, 1995.
3. Ceschi, M., Sillitti, A., Succi, G., Panfilis, S. D.: Project Management in Plan-Based and Agile Companies. *IEEE Software*, Volume 22, No. 3 (2005), 21 – 27.
4. Kadlec, V.: Programujte agilně, nic jiného vám nezbývá! A nebo ano? In: http://www.zive.cz/h/Programovani/AR.asp?ARI=110219, (naposledy aktualizované 8.4.2003).
5. Little, T.: Context-Adaptive Agility: Managing Complexity and Uncertainty. *IEEE Software*,Volume 22, No. 3 (2005), 28 – 35.

6. Patton, J.:Unfixing the Fixed Scope Project: Using Agile Methodologies to Create Flexibility in Project Scope. *In: Agile Development Conference 2003*, p. 146. http://www.abstractics.com/papers/UnfixingScope.pdf
7. Turbit, N.: Defining the Scope of a Project. In: http://www.projectperfect.com.au/downloads/info_define_the_scope.pdf (naposledy aktualizované 27.6.2005)

## Annotation

*Agile software development methods and project scope*

Traditional software development methods are not sufficient in cases of requirements on rapid software development and its repeated changes. Customers require fast delivery with high quality (and appropriate price). Time and quality are variables of project, which are influenced by scope. In the beginning, project scope is fixed, but after some time it started to change (raising), which could have negative influence on success of project. We have to know that project scope changes, in the beginning of project. References used in this paper support a fact that scope managing could be more effective using agile development. This paper deals with question: what should agile methodologies for us, defines project scope and other project variables, shows the more effective way of project scope managing, writes influence of complexity and uncertainty to project. At the end of this paper, tips for help during agile development (in association of project scope) are written.

# Manažment v softvérovom inžinierstve z hľadiska vedenia tímu

# Vývoj tímu v softvérovom projekte a vplyv na manažment

PETER BARTALOS

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
peter.bartalos@stonline.sk

**Abstrakt.** Už veľmi dávno si ľudia uvedomili, že individuálna snaha jednotlivcov dosiahnuť určitý cieľ nie je efektívna. Uvedomili si, že spolupráca s inými pri riešení problémov je obrovským prínosom. Preto ľudia vytvárajú skupiny, v ktorých sú schopní dosahovať omnoho lepšie výsledky ako pri individuálnej činnosti. Táto práca hovorí o procese vývoja tímov a potrebe jeho manažovania. Zaoberá sa fázami, ktorými tímy prechádzajú počas svojej existencie. Hovorí o problémoch, ktoré môžu počas týchto fáz negatívne vplývať na úspech tímu a vysvetľuje potrebu manažovania skupín ľudí.

## Úvod

Ľudia sú spoločenské tvory. Žijú v komunitách s inými ľuďmi. Ich potreba existencie v spoločnosti iných je prirodzená. Táto spolupatričnosť k svojmu druhu sa prejavuje v najrôznejších oblastiach. Ľudia sa narodia do prostredia, kde žije mnoho iných ľudí, vyrastajú v ňom, prežívajú aktívnu časť svojho života a napokon v spoločnosti ľudí aj umierajú.

Táto potreba ľudí byť v spoločnosti iných, sa prejavuje aj v ich práci. Je prirodzené, že skupina ľudí dosahuje omnoho lepšie výsledky pri riešení množstva problémov ako jednotlivci. Avšak viac ľudí si vyžaduje aj kvalitnejšiu organizáciu. Práca väčšieho množstva ľudí sa môže stať chaotickou a môže viesť k ešte horšiemu výsledku ako individuálna činnosť. Preto je potrebné takéto skupiny riadiť, usmerňovať a podporovať ich vývoj a činnosť [1, 3, 4].

## Čo je to tím?

Existuje mnoho definícií pojmu „Tím“. Každá z nich hovorí o určitej skupine indivíduí. Avšak existencia skupiny neznamená, že jej členovia spolu tvoria tím.

*Tím je skupina ľudí, ktorá spolupracuje na splnení určitého spoločného cieľa.*

*Manažment v softvérovom inžinierstve, december 2005, s. 69-75.*

Ako uvedená definícia hovorí, zo skupiny sa stáva tím, keď jej členovia na seba začnú vplývať s cieľom vytvoriť spoločné dielo. Práve väzby medzi členmi, ich spoločná snaha a zodpovednosť robia tím tímom. Zmysel tímu spočíva v [1]:

- *Rýchlosť* – tímová práca je veľmi dobrá pri vývoji určitého produktu, pretože jeho vývoj môže byť paralelný. Členovia tímu rozvíjajú rôzne aspekty projektu v jednom čase.
- *Zložitosť* – tímy zlepšujú schopnosť organizácie riešiť problémy, pretože prinášajú rozličné pohľady, nápady pre proces tvorby.
- *Kreativita* – ľudia z rôznych oblastí sa spoločne zameriavajú na jeden problém.
- *Učenie sa* – členovia tímu sa od seba navzájom učia. Tímová práca zvyšuje produktivitu a poskytuje možnosť pracovať s rozdielnymi ľuďmi.
- *Styčný bod* – tím pôsobí ako styčný bod pre poskytovanie informácií medzi členmi tímu a tvorbu rozhodnutí v tíme.

## Tím sa vyvíja

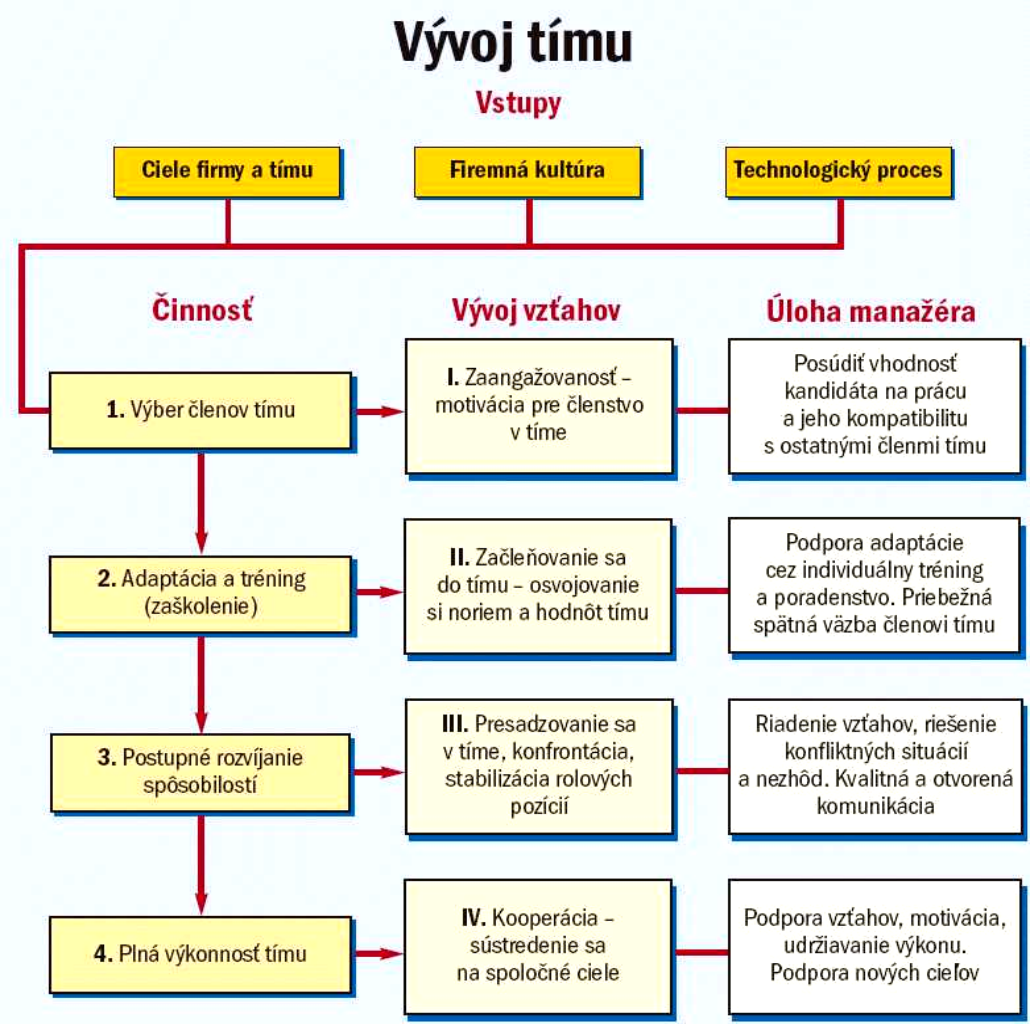
Tak ako každé indivíduum, aj skupina ľudí tvoriacich tím sa vyvíja. Vývoj tímu zahŕňa zvyšovanie schopností jednotlivcov projektového tímu ako aj schopnosti tímu fungovať ako celok [2]. Ide o prirodzený proces, ktorý je však potrebné manažovať. Proces ako taký pozostáva z viacerých fáz, pozri Obr. 1.

Prvou fázou vývoja tímu je výber jeho členov, ktorú má na starosti manažér. Výber by mal zohľadňovať nielen odbornú úroveň potenciálneho člena tímu, ale aj jeho schopnosť tímovej spolupráce. Ide o veľmi citlivú etapu. Zlé zloženie tímu môže projekt, v ktorom je tím angažovaný odsúdiť na neúspech. Ukazuje sa, že dobré obsadenie tímu je omnoho dôležitejšie ako mnoho iných faktorov vplývajúcich na produktivitu tímu [5]. Manažér musí mať dobrý cit pre odhad charakteru ľudí a stále musí mať na zreteli, že ľudia, z ktorých tím vytvorí, budú musieť spolu interagovať. Členov je preto potrebné vybrať tak, aby boli splnené predpoklady pre ich úspešnú spoluprácu.

V ďalšej fáze vývoja tímu dochádza k adaptácii a tréningu jednotlivých členov. V tíme sa stretli rozdielni ľudia, ktorí sa navzájom nemusia vôbec poznať. V tejto fáze sa začínajú spoznávať, zisťujú vlastnosti ostatných členov. Pre vytvorenie dobrého tímu je dôležité, aby jednotliví členovia spolu dobre vychádzali. Dobré vzťahy medzi členmi sú veľmi dôležité [3, 5]. Ich budovanie sa môže realizovať rôznymi aktivitami [6], ktoré nemusia súvisieť s projektom, na ktorom tím pracuje. Dôležité je, aby si členovia tímu uvedomili, že patria do kolektívu. V tejto fáze sa v tíme budujú aj pravidlá, určujú sa hodnoty, ktorými sa bude riadiť.

V nasledujúcej fáze dochádza k rozvíjaniu spôsobilostí členov tímu. Tu sa už členovia začínajú zameriavať na riešenie úlohy. Stabilizujú sa pozície a vzťahy medzi členmi. V tíme môžu vznikať konflikty, ktoré je potrebné riešiť.

Poslednou fázou vývoja tímu je plná výkonnosť tímu. V tejto fáze sa tím sústreďuje na riešenie zadanej úlohy, ktorá je spoločným cieľom pre všetkých členov. Z pohľadu spoločnosti je táto fáza vývoja tímu najdôležitejšia, keďže až tu produkuje tím výsledok, ktorý je pre spoločnosť dôležitý. Manažér musí vytvárať také prostredie pre tím, v ktorom dokážu produkovať čo najlepšie výsledky. Veľmi dôležité je motivovať tím k práci. Členovia tímu musia mať pocit, že ich práca je užitočná. Ciele, ktoré treba splniť nesmú byť nereálne, viedlo by to k demotivácii.




**Obr. 1.** Fázy vývoja tímu (zdroj obrázka [7]).

## Na úspech tímu vplýva množstvo faktorov

Celkový úspech tímu je ovplyvnený veľkým množstvom rôznych faktorov, ktoré sú na sebe často závislé a ich účinok sa spolu zosilňuje. Medzi základné patria:

- Zloženie tímu
- Vedenie tímu
- Komunikácia
- Spolupráca

### Zloženie tímu

Ako som spomínal, skupina ľudí sa stáva tímom vo chvíli, keď ich spoločná snaha vedie k riešeniu spoločného cieľa. Jednotliví členovia tímu preto musia byť schopní spolu existovať. Tím je zložený z rôznych indivíduí, z ktorých každý zaujíma v tíme určité postavenie [1], má svoj charakter, svoje klady a zápory. V tíme je dôležité, aby sa jeho členovia navzájom tolerovali. Je zrejme nemožné nájsť do tímu ľudí, z ktorých každý s každým výborne vychádza. Je prirodzené, že v tíme z času na čas dochádza ku konfliktom. Dôležité je, aby sa tieto konflikty dokázali vyriešiť a aby ich bolo čo najmenej. Musia sa o to snažiť všetci. Budovanie dobrých vzťahov v tíme je veľmi užitočné [3].

### Vedenie tímu

Veľmi dôležitú rolu vo vývoji tímu hrá manažér. Ide o osobu, ktorá by mala usmerňovať celý proces vývoja, vytvárať vhodné prostredie pre tento proces a predvídať a vyvarovať sa situáciám, ktoré by mohli mať negatívne následky. Dobrý manažér by mal:

- Rešpektovať členov tímu
- Mať silné a stabilné postavenie
- Stáť za svojimi rozhodnutiami
- Motivovať členov tímu
- Uľahčovať a spríjemňovať prácu, zabezpečiť vhodné prostredie
- Porozumieť potrebám členov tímu

Manažér je osoba, ktorá je prítomná od úplného začiatku existencie tímu. On je ten, kto vybral členov tímu. Jeho práca je obzvlášť dôležitá a to v každej fáze vývoja. Mala by to byť osoba, ktorá vie porozumieť všetkým členom tímu. Z pohľadu osobnosti by to mal byť skôr extrovert a autoritatívna osoba, ktorú si všetci členovia vážia, ale ak majú problémy, sú schopní sa naňho obrátiť s prosbou o pomoc. Jeho schopnosť predvídať, môže byť nesmierne užitočná. Ak manažér dokáže predvídať rôzne konfliktné situácie, môže usmerniť vývoj situácie. V ideálnom prípade potom konflikt vôbec nenastane. Táto schopnosť môže ušetriť množstvo námahy, ktorú by tím musel vynaložiť a vôbec by neviedla k produkovaniu výsledku.

Manažér tímu nesmie mať slabé postavenie. Ak nie je schopný implementovať a koordinovať pravidlá, na ktorých je tím postavený, môže to mať fatálne následky. Ak sa v tíme so slabým vedením vyskytuje osoba, ktorá nie je schopná podriaďovať sa iným, môže dochádzať ku konfliktom, ktoré majú negatívny vplyv na úspech tímu. Na druhej strane nie je dobré ani to, ak má tím príliš autoritatívne vedenie. Taká osoba sa môže pokúšať vnucovať členom tímu spôsob práce podľa svojich predstáv. Toto však môže viesť k nechuti pracovať a k úplnej demotivácii.

**Komunikácia**

Potreba komunikácie v tíme je samozrejmá [8]. Ak majú členovia tímu spoločne riešiť problém, je nutné aby spolu dokázali komunikovať. Bez vzájomnej otvorenej komunikácie nie je možné hovoriť o tíme.

Vymieňanie názorov, vedomostí vedie k omnoho efektívnejšej práci tímu. Práve vtedy sa uplatňujú výhody tohto spôsobu práce. Každý člen tímu má rôzne schopnosti a vedomosti. Jednotliví členovia by sa mali navzájom dopĺňať a využívať vedomosti ostatných.

Treba si uvedomiť, že nestačí len to, aby členovia spolu hovorili alebo písali si. Nutné je, aby si počas toho navzájom predávali informácie. Členovia preto musia byť schopní vysvetliť svoje myšlienky a ostatní sa musia snažiť ich pochopiť.

Na začiatku, keď sa ešte jednotliví členovia dobre nepoznajú, môže byť komunikácia problémom. Ich vzájomný odstup môže byť prekážkou. Tieto bariéry sa však postupom času odbúravajú. Členovia tímu sa spoznávajú. Môže medzi nimi vzniknúť aj kamarátsky vzťah, čo je veľmi veľkým prínosom. Ich vzájomné vzťahy nemusia byť spojené len s tímom, môžu sa týkať aj rôznych mimopracovných aktivít. Tie veľmi prispievajú k schopnosti členov navzájom spolu komunikovať. Vedie to k lepšiemu spoznaniu a keď niekoho lepšie poznáme, vieme ho aj lepšie pochopiť. V tíme je vhodné hovoriť aj o vzájomných vzťahoch. Tomuto sa väčšinou nevenuje veľká pozornosť. Avšak takáto diskusia môže viesť k jednoduchému riešeniu množstva problémov.

**Spolupráca**

Efektivita práce v tíme je veľmi úzko spätá so schopnosťou jednotlivých členov spolupracovať s ostatnými. Nefungovanie tímovej spolupráce môže mať veľmi vážne následky. Podľa zistení 30 až 80 percent času stráveného nad spoločným riešením problému sa využíva neefektívne [7]. Mark Klein, zaoberajúci sa konfliktami pri projektovaní zložitých zariadení v štúdii pre americký MTI konštatuje, že až 50 percent interakcií medzi projektantmi obsahuje konflikt. Kľúčovým faktorom efektívnosti procesov projektovania zložitých zariadení, je podľa neho práve schopnosť efektívne riešiť konfliktné situácie.

Tímová spolupráca spočíva v rozdelení si úloh. Každý člen rieši čiastkový podproblém, pričom na jednom podprobléme môže spolu pracovať viacej členov. Následne, výsledky jednotlivých členov prispievajú k riešeniu celkového problému. Túto prácu je potrebné organizovať. Jednotliví členovia musia vedieť o práci ostatných členov.

V tímoch, kde je v značnej miere potrebná spolupráca jednotlivých členov nie sú vhodní rivalitní jedinci, aj keď to môžu byť veľmi dobrý odborníci. Takýto ľudia sú vhodní do tímov, kde spolupráca nie je až tak potrebná. Väčšina projektov si však vyžaduje úzku spoluprácu v tíme. V nich je potrebné, aby členovia uznávali prácu ostatných.

## Záver

Vývoj tímu je prirodzený proces prebiehajúci v každom zoskupení ľudí. Tento proces je potrebné riadiť.

V tejto eseji som ozrejmil potrebu riadenia vývoja tímu. Hovoril som o faktoroch, ktoré naň vplývajú. Spomenul som dôležitosť dobrého zloženia tímu, ktorý musí byť vedený kvalitným manažérom. Vysvetlil som potrebu komunikácie a spolupráce v tíme. Všetky tieto faktory sa navzájom ovplyvňujú. Preto je dôležité, aby členovia tímu čo najviac dbali na dodržiavanie všetkých zásad, ktorými sa treba v tíme riadiť.

## Použitá literatúra

1. Barnum C. M.: *Building a team for user-centered design*. In Proceedings of IEEE Professional Communication Society international Professional Communication Conference and Proceedings of the 18th Annual ACM international Conference on Computer Documentation: Technology & Teamwork (Cambridge, Massachusetts, September 24 - 27, 2000). ACM Special Interest Group for Design of Communications. IEEE Educational Activities Department, Piscataway, NJ, 325-332.
2. Bieliková M.: *Softvérové inžinierstvo – Princípy a manažment*. Slovenská technická univerzita v v Bratislave, 2000.
3. Brereton O.P., French A., Layzell P.: *Supporting collaboration in distributed software engineering teams.* apsec, p. 38, Seventh Asia-Pacific Software Engineering Conference (APSEC'00), 2000.
4. Ferrara J. C.: *Building positive team relationships for better usability*. interactions 12, 3 (May. 2005), 20-21. DOI= http://doi.acm.org/10.1145/1060189.1060207
5. Gorla N., Lam Y. W.: *Who should work with whom?: building effective software project teams*. Commun. ACM 47, 6 (Jun. 2004), 79-82. DOI= http://doi.acm.org/10.1145/990680.990684
6. Infoart. *Outdoor-training*. (prospekt v nemeckom jazyku) http://www.infoart.ch/deutsch/managementtraining/outdoor/index.html
7. Kališ M.: *Tímová spolupráca vo firme*. Hospodárske noviny, 2005. http://hnonline.sk/?s1=k&s2=0&s3=4&s4=1&s5=0&s6=0&m=detail&article[area_id]=10025630&article[id]=22761215&p=k04100_detail

8. Michailidis A., Rada R.: *Organizational Roles and Communication Modes in Team Work.* hicss, p. 8068, 34th Annual Hawaii International Conference on System Sciences ( HICSS-34)-Volume 8, 2001.

## Annotation

*Team building in software projects and the impact on management*

People have realized a long time ago, that an individual approach to achieve a united goal, is not effective enough. They have realized that collaboration with others brings many benefits. This causes people create groups in which they solve problems together. This essay deals with team building and the associated need for its management. It analyses the phases of team building and other team related problems.

# Komunikácia v tíme

## Lokalizovaný vs. distribuovaný tím

MARTIN ADAM

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
martin@atrip.sk

**Abstrakt.** Väčšina ľudí pracuje v prostredí, kde je potrebné spolupracovať s inými ľuďmi a každá forma spolupráce si vyžaduje komunikáciu v nejakej forme. Z pohľadu typu komunikácie nie je dôležité, či je tím lokalizovaný alebo distribuovaný. Vždy sa budú diskutovať technológie, možné riešenia úloh, ale aj viesť osobné rozhovory. Otázkou zostáva, aké prostriedky sa na to použijú, nakoľko je použitá metóda komunikácie efektívna a kedy ktorý typ komunikácie zvoliť. Odpoveď na tieto otázky do značnej miery závisia na tom, či, a ako často, sa ľudia spolupracujúci na nejakom projekte stretávajú.

## Komunikácia

V každom pracovnom tíme, a to nie len softvérovom, je veľmi dôležitou podmienkou progresu fungujúca komunikácia medzi jeho členmi. Bez nej by sa asi aj ťažko dalo hovoriť o tíme. Komunikácia má takisto značný dosah na výkon pracovníkov v tíme a významne môže prispievať k pocitu zadosťučinenia [4].

Spolupráca ľudí v tíme môže prinášať problémy pri komunikácii aj v malých tímoch, nehovoriac o väčších, kde je potrebné komunikáciu už aj rozumne riadiť. Toto všetko sa však stáva ešte zložitejším pri distribuovaných tímoch. A práve informatika je odvetvie, kde sú distribuované, resp. virtuálne, tímy veľmi bežným javom. V takejto spolupráci sa neraz ocitnú aj ľudia z rozdielnych krajín a kultúr, čo tím môže obohatiť ale takisto to môže viesť k nedorozumeniam spôsobeným kultúrnou alebo jazykovou rozdielnosťou.

V tomto dokumente sa budem snažiť čitateľa vniesť do problematiky manažovania komunikácie v tíme a pozastaviť sa nad rozdielmi medzi tímom, ktorý je na spoločnom mieste s možnosťou reálne sa stretávať (lokalizovaný tím) a tímom, kde členovia tímu sú na rozdielnych miestach (distribuovaný / virtuálny tím), čiže sú odkázaní na použitie pomocných prostriedkov pri komunikácii.

*Manažment v softvérovom inžinierstve, december 2005, s. 76-81.*

## Typy komunikácie

Existuje viacero kritérií, podľa ktorých môžeme rozdeliť komunikáciu do niekoľkých skupín. Jedno z delení je do nasledujúcich dvoch dvojíc [4]. Každá komunikácia pritom spadá vždy naraz do jednej kategórie z prvej dvojice a jednej z druhej, čím vznikajú štyri možné kombinácie (zaradenia):

- Formálna
- Neformálna
- Horizontálna
- Vertikálna

Formálna komunikácia predstavuje tradičný prístup, kde manažér, nadriadený alebo iný pracovník vo vedúcej úlohe, formálne smeruje inštrukcie smerom k zamestnancom. Takýto typ môžeme zároveň považovať za vertikálny smerom dole. Na efektívnosť tímu tu výrazne vplýva jasnosť informácií nadriadených podriadeným. Za formálnu sa môže považovať ale aj komunikácia pracovníkov na rovnakej úrovni (napríklad vedúci oddelení) a v takomto prípade hovoríme o horizontálnej formálnej komunikácii.

Neformálnou komunikáciou môžeme nazvať väčšinu rozhovorov medzi zamestnancami navzájom. Napríklad aj spoločné obedy alebo športové aktivity. Tento typ komunikácie pomáha zdvíhať morálku a produktivitu pracovníkov [4] tým, že sa spoznávajú. Tento typ je v prevažnej miere horizontálny, čo však nie je podmienkou.

Horizontálny typ zahŕňa komunikáciu kolegov. Efektívna horizontálna komunikácia môže napomôcť vytvoriť uvoľnenú atmosféru a tým zvýšiť produktivitu. Táto komunikácia prebieha primárne neformálne, ale môže aj formálne.

Vertikálna komunikácia vzniká medzi ľuďmi vo vzťahu nadriadený / podriadený. Prebieha oboma smermi (smerom dole aj hore), ale väčšinou prevažuje komunikácia smerom nadol. Vo veľkej prevahe sa jedná o formálnu komunikáciu.

Pri dorozumievaní sa v rámci tímu, najmä ak to je menší tím (napríklad tím študentov), sú používané všetky vymenované druhy komunikácie, ale napriek tomu je vertikálna využívaná veľmi obmedzene (ak vôbec). Väčšinou iba ak existuje explicitný vedúci tímu. Silno vertikálna komunikácia v takomto prípade predstavuje hlavne komunikáciu medzi tímom a nadriadeným (učiteľom).

## Prostriedky komunikácie

Doteraz vymenované typy komunikácie platia pre akýkoľvek tím, nijak nezáleží na tom, či sa členovia tímu alebo ich nadriadený nachádza v tej istej miestnosti, budove, meste alebo štáte. To, ako je tím distribuovaný už ale má veľký vplyv na to, aké prostriedky a v akej miere sa pre dorozumievanie používajú.

Pri distribuovanom tíme sú účastníci odkázaní prakticky len na technické prostriedky, prevažne IT techniku, ktorá im umožňuje navzájom komunikovať. Reálne stretnutia nie sú vylúčené, ale aj keď sa vôbec konajú, je to zriedkavé a nemusia sa ich zúčastniť všetci členovia.

Pri lokalizovanom tíme, ktorý je v ideálnom prípade na rovnakom pracovisku, alebo sa je aspoň schopný veľmi často stretávať, je situácia iná. Členovia sa môžu rozprávať tvarou v tvár, diskusia prebieha živšie a dynamickejšie a na problémy sa dá

reagovať okamžite. Takéto tímy tiež zvyčajne používajú tie isté prostriedky komunikácie ako distribuované tímy, ale slúžia hlavne na doplnenie a obohatenie reálneho rozhovoru a urýchľujú prácu [3] (napríklad zdieľaním dokumentov). Takisto je vďaka ním často možné najmä vo väčších tímoch skrátiť trvanie schôdzí resp. znížiť ich pravidelnosť, čím sa šetrí čas pracovníkom.

Medzi najčastejšie používané IT prostriedky komunikácie patrí:

- Telefónny rozhovor / konferencia (či už klasický alebo po internete)
- Video konferencia (obohatenie o neverbálne prvky)
- Zverejňovanie na webe
- Diskusné fóra resp. news servery
- Bežné emaily
- Diskusie v reálnom čase (napr. IRC)
- Instant messaging (napr. ICQ)
- Rôzne úložiská resp. CVS systémy

Použitie konkrétneho prostriedku závisí od niekoľkých skutočností. Jednou z nich sú určite aj technické možnosti. Ak členovia tímu nedisponujú rýchlym pripojením do internetu, asi nebudú používať video konferenciu, a dokonca ani audio konferencia nemusí mať požadovanú kvalitu.

Ďalšou dôležitou vecou je, či je tím lokalizovaný alebo nie. Ak je pre členov tímu jediná možná komunikácia na diaľku, určite budú používať viac diskusie v reálnom čase, konferenčné diskusie a podobne, kým lokalizovaný tím bude vo väčšej miere (v pomere k ostatným technickým prostriedkom) používať emaily alebo úložiská a komunikácia v reálnom čase bude prebiehať prevažne priamo na pracovisku.

## Typy komunikácie II.

Využitie konkrétnych prostriedkov komunikácie závisí ale aj od toho, čo sa chce docieliť. Napríklad ak sa tvorí dokumentácia, bude potrebné vytvoriť nejaký priestor (úložisko, CVS systém apod.), kde sa bude dať dodávať stále najnovšia verzia a sledovať kto aké zmeny spravil. Ak sa naopak riešia problémy, kde treba rýchlo vyriešiť vzniknutú situáciu a je nutné do toho zapojiť viacero členov tímu, môže byť konferenčný hovor alebo virtuálna diskusná skupina (napr. na IRC) dobrou voľbou.

Podľa toho, aký je cieľ našej komunikácie, mení sa aj jej typ. Z tohto hľadiska existujú dva základné typy [3]:

- Synchrónna
- Asynchrónna

Do synchrónnej patrí už spomenutý telefónny rozhovor, resp. aj video konferencia, diskusie v reálnom čase (tzv. „chatovanie“) a samozrejme reálny rozhovor ľudí, ktorý však z pochopiteľných dôvodov pri distribuovaných tímoch nie je možný. Pri synchrónnej komunikácii je jednou z podstatných výhod, že je možné okamžite

reagovať na situácie a aj spätná väzba je veľmi rýchla. Nevýhodou je, že ľudia musia byť „zosynchronizovaní“ – v tom istom čase na tom istom mieste (minimálne virtuálne). Je to ale veľmi dobrý spôsob na rýchle riešenie problémov. K takejto komunikácii patrí pri lokalizovaných tímoch aj stretnutie, ktoré so sebou prináša napríklad aj výhodu toho, že vždy vieme, či máme pri rozhovore naozaj plnú pozornosť poslucháča. Zo skúsenosti môžem povedať, že ak má tím možnosť takýchto stretnutí, je komunikácie pomocou IT prostriedkov podstatne nižšia a niektoré sa používajú v minimálnej miere (ak vôbec). Takisto sa ľudia pri virtuálnych stretnutiach majú tendenciu správať formálnejšie ako v reálnom styku.

Typickou, a veľmi častou, asynchrónnou komunikáciou je email. Ale patrí sem aj zverejňovanie údajov na webe alebo v úložiskách. Z vlastnej skúsenosti môžem povedať, že využitie takýchto prostriedkov je veľmi praktické bez ohľadu na to, či je tím distribuovaný alebo nie, pretože je vhodným doplnkom aj pre lokálne tímy. Dokáže podstatne pomôcť pri vyvíjaní softvéru aj dokumentácie. Emaily sú vhodné na manažovanie tímu, keď sa dohodujú stretnutia, pripomínajú sa termíny a podobne. To, čo sa povie (aj na „chate“), sa môže zabudnúť. Ale email sa vo veľa klientoch dá označiť ako dôležitý a človek ho má „na očiach“. Výhodou asynchrónnej komunikácie je aj možnosť zdieľania súborov (či už v úložisku alebo príloha emailu). Takisto nie je potrebné, aby boli členovia tímu v rovnakom čase na rovnakom mieste (aj keď to miesto môže byť len virtuálne rovnaké, musia sedieť naraz všetci pri počítači / telefóne) Na druhej strane zas nie je k dispozícii okamžitá reakcia a takýmto spôsobom sa ťažšie diskutuje problém. Najmä kombinácia email a príliš veľa členov tímu v diskusii môže byť neprehľadná. V takomto prípade je vhodnejšie použiť fórum alebo news server a ešte lepšie sa stretnúť (ak to je možné).

Instant Messaging (ICQ, Jabber, apod.) patrí v podstate do oboch kategórií, nakoľko umožňuje aj synchrónnu aj asynchrónnu komunikáciu (aj keď tá je v prípade ICQ trochu obmedzená). Je to akýsi hybrid medzi týmto dvoma spôsobmi komunikácie.

Ukážkou spolupráce na diaľku je aj študovanie na diaľku. Výskumom [3] sa zistilo, že vhodným pomerom pri vyučovaní na diaľku je pre používanie synchrónnej formy k asynchrónnej pomer 20 / 80, kedy sa začínajú dosahovať lepšie výsledky, ako keby mali členovia tímu (v tomto prípade študenti) riešiť všetko formou asynchrónnej komunikácie.

## Diskutované témy

Aj keď skladba komunikácie je pri lokalizovaných a distribuovaných tímoch rozdielna, diskutované témy sú tie isté, len intenzita ich diskutovania môže byť rozdielna. Členovia lokalizovaného tímu majú určite oveľa viac príležitostí na neformálnu komunikáciu a asi sa aj oveľa rýchlejšie spoznajú. Nakoľko ale ľudia v distribuovanom tíme majú veľmi obmedzené možnosti sa spoznať, môžu byť najmä počiatočné fázy tvorby tímu ťažké [2], kedy je potrebné si nájsť miesto v tíme a vytvoriť prirodzené štruktúry.

Napriek tomu všetkému, vždy sa diskutuje o tých istých veciach. O manažmente tímu, použitých a potenciálnych technológiách, aké nástroje a ako použiť, ale vedú sa aj osobné rozhovory [3]. Podľa témy sa volia aj prostriedky na komunikáciu, resp. podľa práve používaného prostriedku sa mení skladba rozhovoru. Napríklad pri použití asynchrónnych metód komunikácie (napr. email) sú členovia tímu oveľa vecnejší a formálnejší ako pri synchrónnej diskusii (napr. fórum alebo Instant Messaging), kde môže miera osobných rozhovorov dosiahnuť aj štvrtinu celkového objemu hovoru [3].

## Rozmanitosť skupiny – prínos alebo problém?

Pri tvorbe tímu sa doň často dostanú rozmanití ľudia, ktorí môžu byť aj z dosť rozdielnych prostredí. Takáto situácie môže nastať najmä pri distribuovaných skupinách veľmi ľahko. Rozdiely medzi členmi tímu môžeme rozdeliť do dvoch základných skupín [1]:

- S vysokou viditeľnosťou
- S nízkou viditeľnosťou

Vysokú viditeľnosť má najmä rasa, pohlavie a vek člena tímu. Nízkou viditeľnosťou sa vyznačujú najmä postoje člena a jeho hodnoty, jeho vzdelanie, doterajšie skúsenosti, získané zručnosti a znalosti.

Aj keď distribuované tímy sú považované za trochu menej efektívne [1], do značnej miery odstraňujú rozdiely členov s vysokou viditeľnosťou, nakoľko túto viditeľnosť odstraňuje samotný charakter distribuovaného tímu.

Pri členoch z rôznych kultúr môžu nastať problémy, ktoré zvyknú vznikať aj kvôli jazykovými nedorozumeniam, na druhej strane rôznorodosť podporuje kreativitu tímu [1] a vnáša viacero pohľadov na problémy, čo môže byť veľkým prínosom.

## Ktorý typ tímu je teda lepší?

Z pohľadu efektivity práce by bola asi odpoveď na otázku položenú v nadpise pomerne jednoznačná. Tím, ktorý pracuje na jednom mieste a má možnosť sa stretávať, v ideálnom prípade dokonca pracovať na jednom mieste, je efektívnejší [1], rieši problémy rýchlejšie a vyžaduje menej manažovania a koordinovania [2]. Napríklad ak tím pozostáva z členov v rozdielnych krajinách a je potrebné sa stretávať synchrónne, vznikajú problémy s pracovným časom (rôzne časové pásma) a s pracovnými dňami (rôzne krajiny majú rôzne sviatky, ale aj rôzne voľné dni v týždni – napríklad arabské krajiny).

Otázka v reálnom živote ale asi takto nestojí, pretože distribuované (virtuálne) tímy nevznikajú kvôli tomu, že by boli efektívnejšie, ale preto, že to je častokrát lacnejšie a pohodlnejšie riešenie, resp. v tíme je potrebný človek alebo ľudia, ktorí sú od seba tak vzdialení, že pravidelné a časté stretávanie je nemožné.

Takže to, z akých členov bude tím nakoniec zložený, závisí od manažéra projektu, ktorý si musí zvážiť, či sú riziká distribuovaného tímu vyvážené jeho prínosom, či už

finančným alebo v ľudských zdrojoch, ktoré je možné takto nasadiť. V každom prípade dnešné technológie už ponúkajú dostatočné prostriedky na to, aby distribuovaný tím mohol riadne fungovať a produkovať požadované výstupy v požadovanej kvalite a požadovanom čase. Technológia je však len prostriedok na komunikáciu a iba čiastočná náhrada reálnych stretnutí. V žiadnom prípade to nie je riešenie alebo záruka fungovania spolupráce v tíme. Je to len pomôcka, ktorou sa úloha výrazne uľahčuje.

## Použitá literatúra

1. Beise, C. M..: IT project management and virtual teams. *Proceedings of the 2004 SIGMIS conference on Computer personnel research: Careers, culture, and ethics in a networked environment* (2004) 129-133 (anglicky).
2. Brandow, D.: Geographically distributed work groups and IT: a case study of working relationships and IS professionals. *Proceedings of the 1997 ACM SIGCPR conference on Computer personnel research* (1997) 87-92 (anglicky).
3. Graveline, A., Geisler, C., Danchak, M.: Teaming Together Apart: Emergent Patterns of Media Use in Collaboration at a Distance. *Proceedings of the 18th annual ACM international conference on Computer documentation: technology & teamwork* (2000) 381-393 (anglicky).
4. Javed, T., Maqsood, M., Durrani, Q. S..: A survey to examine the effect of team communication on job satisfaction in software industry. *ACM SIGSOFT Software Engineering Notes*, (2004) Volume 29, Issue 2, 6-6 (anglicky).

### Annotation

*Team communication – localized vs. distributed team*

Most people work in their jobs or schools in environment, where they have to cooperate with other persons and every such cooperation implies some sort of communication between them. There is no big difference between localized and distributed team in type of communication of its members. They will always discuss technologies, possible solutions to problems, but also to take part on personal talk. However, the question is, what means they will use to communicate to each other, and how good and efficient is the chosen method of communication. The answer to this question depends greatly on how often, if at all, meet the people working in team on some project.

# Manažment konfliktov v tíme

MICHAL BARLA

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
barla01@student.fiit.stuba.sk

**Abstrakt.** V každom tíme zloženom z navzájom interagujúcich ľudí, ktorí majú za cieľ niečo spoločné vytvoriť, raz príde ku konfliktu. Tento konflikt môže byť pre tím konštruktívny, ale aj deštruktívny. Navyše aj konštruktívny konflikt, ak nie je riadený, môže eskalovať do konfliktu deštruktívneho. To môže mať fatálne dôsledky na výstup projektu a na tím samotný. V tejto práci sa zaoberám rôznymi typmi konfliktov vyskytujúcich sa v tíme, dôvodmi ich vzniku a možnosťami, ako ich účinne riadiť. Úlohu manažéra pri riešení konfliktov vysvetľujem na príklade softvérového tímu a na konfliktoch medzi vývojármi softvéru a testermi. Práca nazerá na zdroje konfliktov týchto skupín z viacerých hľadísk a naznačuje potrebné kroky zo strany manažéra vedúce k ich riešeniu.

## Úvod

Každý tím je tvorený z unikátnych jedincov. Na svete zrejme neexistujú dvaja úplne rovnakí ľudia. Táto unikátnosť spolu s nutnosťou vzájomnej interakcie v tíme má na svedomí vznik konfliktu medzi jednotlivými členmi tímu. Konflikt je často podmienený osobnostnými charakteristikami účastníkov konfliktu. Predpoklady na vznik konfliktu však môžu vychádzať aj z pracovnej pozície jednotlivcov v tíme.

Dobrý manažér tímu by nemal nechať konflikty v tíme nepovšimnuté, neriadené ak mu záleží na tom, aby tím vyprodukoval kvalitný výsledok a aby sa jednotliví členovia v tíme cítili dobre.

Ak chceme čokoľvek úspešne riadiť, musíme to dobre poznať. Preto v tejto eseji definujem čo je konflikt a rozoberám jeho potenciálne zdroje a spôsoby jeho riadenia. Špecificky sa zameriavam na softvérové tímy, v ktorých je typický konflikt medzi vývojármi a testermi. Riadenie konfliktov v tíme vysvetľujem práve na sporoch medzi týmito dvoma dôležitými súčasťami každého dobrého softvérového tímu.

## Konflikt

Konflikt je stav odporu, nesúhlasu alebo nekompatibility medzi dvoma a viacerými ľuďmi či skupinami, ktorý je v niektorých prípadoch charakterizovaný psychickým

*Manažment v softvérovom inžinierstve, december 2005, s. 82-88.*

alebo fyzickým násilím. Všeobecne konflikt vzniká z rozdielnych postojov, názorov, potrieb, hodnôt alebo cieľov dvoch účastníkov konfliktu.

To, aký má konflikt priebeh je podmienené skúsenosťami, výchovou, komunikačnými schopnosťami a psychickou stabilitou zúčastnených strán. Ak by bol uvedený zoznam úplný, mohol by som v tomto okamihu prestať písať, pretože ani jedna z vymenovaných vecí nám nedáva veľa možností ako ju účinne riadiť. Našťastie má na priebeh konfliktu významný dosah aj prostredie v ktorom sa konflikt odohráva a to je jedna z vecí, ktoré môže manažér veľmi úspešne ovplyvňovať. Pod prostredím nemyslím len fyzické prostredie, ale chápem ho v širšom význame.

**Delenie konfliktov**

Konflikty môžeme deliť podľa typu na: [3]

- poznávacie – zamerané na problémy, procesy, princípy;
- citové – zamerané na ľudí, emócie.

Na prvý pohľad je zjavné, že poznávací konflikt je konštruktívny, rieši problémy, zatiaľ čo citový konflikt je deštruktívny, zvyčajne nič nerieši, zbytočne oberá ľudí o energiu a znižuje morálku.

Často sa používajú aj názvy funkčný a dysfunkčný konflikt. Funkčný konflikt, ktorý prináša pozitívne výsledky môže ľahko eskalovať do dysfunkčného konfliktu, ktorý rozruší vzťahy, naruší pracovný proces. Tieto poznatky nám dávajú silnú motiváciu, prečo by mal byť konflikt vždy riadený.

**Stratégie riadenia konfliktov**

Stratégie, ktorými môže postupovať tímový manažér pri kontrole konfliktov, sa delia na dve základné skupiny: [3]

- preventívne stratégie;
- reaktívne stratégie.

Preventívne stratégie sa uplatňujú ešte predtým ako vznikne konflikt. Zahŕňajú zadefinovanie pevných pravidiel pre tímové procesy a pravidiel správania sa tímu v rôznych situáciách. Do tejto kategórie by som zaradil aj všetky aktivity, ktoré majú za cieľ pripraviť tím na konflikt, prípadne znížiť dopady budúceho konfliktu. Opakom preventívnych stratégií sú reaktívne stratégie, ktoré nastupujú keď problém nastane. Príkladom reaktívnej stratégie môže byť využitie autority manažéra, ktorý tímu jasne oznámi svoje rozhodnutie o riešení a očakáva akceptovanie zo strany členov tímu. Ďalším príkladom môže byť hľadanie kompromisu medzi konfliktnými názormi.

Podľa [3] existuje aj systematický prístup k riešeniu konfliktov nazvaný „*krok za krokom*“ (anglicky step-by-step). Keďže sa rieši konflikt, ktorý nastal, ide o reaktívnu stratégiu:

1. Nastavte prostredie tak, aby si všetky strany uvedomili potrebu vyriešiť problém.
2. Uistite sa, že všetky strany ho chcú vyriešiť.

3. Všetky strany musia akceptovať konflikt ako spoločný problém – nie v štýle výhra/prehra.
4. Preskúmajte dôvod konfliktu.
5. Vygenerujte možnosti riešenia.
6. Zúčastnené strany sa musia zhodnúť na jednom z riešení ako na najvhodnejšom.
7. Implementujte riešenie.
8. Overte úspešnosť riešenia.
9. Oslavujte alebo sa vráťte na krok 6.

Tento prístup podľa mňa vo svojich krokoch ukrýva niekoľko pascí. Napríklad splnenie bodu číslo jedna môže byť veľmi náročná a navyše ťažko merateľná úloha. Bod číslo dva postupu nedefinuje, čo v prípade, že jedna zo strán nemá záujem riešiť konflikt (aj keď si môže uvedomovať potrebu ho vyriešiť, nemusí byť táto potreba akútna). Tento prístup k riešeniu konfliktov teda sám o sebe určite nestačí na zvládnutie ľubovoľného problému.

Podľa môjho názoru by sa správny manažér určite nemal spoliehať iba na reaktívne stratégie, ale mal by venovať dostatok pozornosti aj preventívnym stratégiám. Ak bude svoju prácu robiť dobre, nemusí k vážnym deštruktívnym konfliktom vôbec dôjsť. Nie je schopný ten, kto vie dobre „hasiť“ vzniknuté problémy (aj to sa cení), ale ten, kto vie problémom účinne predchádzať.

## Konflikt v tíme

### Vývoj tímu

Podľa Karen Mackey [2] musí prejsť každé zoskupenie ľudí počas svojho vývoja konfliktom. Otázkou je iba ako sa s ním vysporiada. Spory prichádzajú najmä v druhom vývojovom stupni tímu – kryštalizácii (anglicky storming). Snahy vyhýbať sa konfliktu a neriešiť ho zastavia tím v tomto stupni vývoja. Konflikt je v tejto fáze pre tím nevyhnutný, pretože počas neho ľudia pochopia svoje odlišnosti a začnú ich akceptovať. Získajú dôveru v tím a jeho schopnosť riešiť problémy. Zistia, že môžu uviesť problém bez toho, aby pri tom museli niekoho obviniť alebo napadnúť. Naučia sa navzájom počúvať bez potreby sa neustále brániť. Takéto úspešné vyriešenie konfliktu posunie vývoj tímu do ďalšieho vývojového stupňa smerom k dobre stmelenému fungujúcemu tímu. Konflikt vo vyvíjajúcom sa tíme je teda v zásade žiaduci a v prípade, že ho tím prekoná, má veľmi výrazný pozitívny efekt v zmysle hesla „*čo nás nezabije, to nás posilní*“.

Tento scenár vývoja tímu podľa mňa veľmi dobre funguje pre menšie tímy, ale nemusí platiť pri tímoch s väčším počtom členov. Súvisí to s povahou ľudí. Ľudia majú tendenciu vytvárať skupinky, čo sa môže prejaviť aj vo väčšom tíme. Jednotlivec potom skôr preferuje riešenie svojich problémov v rámci svojej skupinky. Tento stav teoreticky môže byť priaznivý, ak sú skupinky vytvorené tak, že kopírujú pracovné skupiny v tíme. Ale v prípade, že sú skupinky vytvorené naprieč celým tímom, nemusí

dôjsť k stavu, keď jedinec získa dôveru v tím. Môže veriť iba svojej skupinke, ktorá však často nedokáže účinne riešiť všetky jeho problémy.

Ďalšou nepríjemnosťou, vyskytujúcou sa v spoločenstvách ľudí je identifikovanie a izolovanie jedinca, ktorý sa určitým spôsobom výraznejšie odlišuje od ostatných. Je zaujímavé, že spoločenstvo si takmer vždy dokáže takéhoto jedinca nájsť. Tento jav sa dá pozorovať už od základnej školy a bohužiaľ jej vychodením nemusí zaniknúť. Potom záleží na vyspelosti členov tímu, či takémuto človeku bude vôbec umožnené efektívne v tíme pracovať a fungovať.

Úlohou manažéra je sledovať a vyhodnocovať správanie sa členov tímu, budovať v nich pocit spolupatričnosti a vzájomnej dôvery. Každý člen tímu by si mal uvedomovať svoju vlastnú užitočnosť pre tím a užitočnosť ostatných. To sa dá docieliť vzájomným informovaním sa o úlohách jednotlivých členov a o postupe prác.

### Zdroje konfliktov v tíme

Problémy môžu nastať aj v tíme, ktorý prekonal počiatočné fázy vývoja. Cohen et al. [1] definuje ďalšie, pre tím špecifické dôvody vzniku konfliktu, ktorými sú:

- obmedzené zdroje;
- vzájomná závislosť úloh;
- rôzna povaha jednotlivých úloh;
- konkurenčný boj o odmeny;
- pocity nespravodlivosti;
- zlé rozdelenie právomocí.

Tento zoznam nie je na prvý pohľad rozsiahly, ale ak si uvedomíme nakoľko obšírne sú jednotlivé pojmy a koľko konkrétnych problémov pokrývajú, tak sa dostaneme k značnému počtu problémov, ktoré na tím striehnu. Napríklad obmedzené zdroje môžu vyjadrovať nedostatok času, financií či ľudí na projekte. Pocity nespravodlivosti zase môžu prameniť zo zlého naplánovania úloh či zlého určenia priorít.

Intuitívne cítime, že väčšinu z uvedených zdrojov môže významne ovplyvniť manažér. Problematické však zostávajú dva body: rôzna povaha jednotlivých úloh a vzájomná závislosť úloh. Bez toho, aby manažér zmenil povahu úloh, tieto problémy priamo nevyrieši. V ďalšej časti sa teda pozrieme najmä na tieto dva dôvody vzniku konfliktu, ktoré preskúmame na príklade softvérového tímu.

### Konflikt v softvérovom tíme

Softvérový tím je špecifickým zoskupením ľudí, ktorí, sú dosť úzko špecializovaní, keďže vykonávajú značne odlišné činnosti. Tieto činnosti na sebe istým spôsobom závisia, avšak ich ciele môžu byť navzájom v protiklade (vývojár má vyprodukovať kód bez chýb, tester má naopak nájsť chyby).

Môžeme teda skonštatovať, že aj v softvérovom tíme, ktorý prekonal fázu kryštalizácie, stále existujú predpoklady na vznik konfliktu. Najčastejšie vyskytujúcim sa prípadom je spor medzi vývojármi a testermi. Zdroj konfliktu je prirodzený: ich úlohy sú na sebe závislé a pritom prirodzene protichodné.

Cohen et al. [1] kategorizuje zdroje konfliktov medzi vývojármi a testermi do troch vrstiev:

1. proces testovania softvéru;
2. ľudia;
3. organizácia.

V každej z týchto vrstiev sa dajú identifikovať možné zdroje konfliktu a následne možnosti podpory riešenia zo strany manažéra.

*Proces testovania softvéru*

Jeden z najčastejšie sa vyskytujúcich zdrojov konfliktu medzi vývojármi a testermi vyplýva z nedostatku časových zdrojov. Veľmi často je dátum dodania produktu pevne stanovený a v prípade omeškania sa najčastejšie ušetrí čas na pôvodne plánovanom testovaní. Testerom teda často ostane príliš málo času na ich prácu, keďže testovanie môže začať až vtedy, keď je čo testovať – keď vývojári dodajú produkt. Dochádza k súpereniu o čas, potrebný pre vykonanie práce. Problém je posilnený aj tým, že vývojári, keďže nie sú na konci projektového snaženia, nemajú rovnaký cit pre dodržiavanie termínov ako testeri. Vývojári sa predsa môžu spoliehať na dobehnutie časového sklzu v ďalšej etape – v testovaní. Z ich pohľadu sa môže zdať časový sklz oprávnený – veď pracujú na tom, aby bol produkt lepší, čím uľahčia a urýchlia prácu testerom.

Povinnosťou manažéra, ktorý chce mať na výstupe dobre otestovaný produkt, je vyhradiť dostatočný čas v pláne na testovanie a zabrániť postupnému ukrajovaniu z tohto času oneskoreniami v predchádzajúcich etapách projektu. Jedným z riešení je plán, počítajúci s oneskoreniami, ktorý obsahuje určitý čas navyše, z ktorého môžu pri oneskorení čerpať všetky etapy projektu. Dôležité je, aby si všetci zainteresovaní uvedomili, že „navyše“ je len tento čas, nič viac.

Iným zdrojom konfliktov vo vrstve procesu testovania softvéru je nepochopenie požiadaviek, resp. rozdielne chápanie požiadaviek na softvér zo strany vývojára a zo strany testera. Zatiaľ čo vývojári sú viac sústredení na technické nuansy riešenia, testeri sa sústreďujú na požiadavky používateľa. Tieto dva rozdielne pohľady môžu často spôsobiť konflikt, keď napríklad vývojár vymyslí nový prístup k získaniu želaného výsledku, ktorý aj keď veľmi dobre fungujúci nemusí byť kompatibilný s aktuálnym vybavením používateľa.

Takéto zdroje konfliktov sa dajú eliminovať jasným definovaním cieľov a zavedením motivácie v tíme pomocou určenia metrík na meranie výkonnosti v procese vývoja a kvality, ktorú treba dosiahnuť.

*Ľudia*

Množstvo konfliktov medzi vývojármi a testermi vzniká na základe ich rozdielnych osobnostných charakteristík a rozdielneho spôsobu myslenia.

Vývojári vyvíjajú softvér tak, aby sa dal použiť na to, načo je určený. Nechápu, prečo by sa mal niekto pokúšať používať ho iným spôsobom. Testeri majú naopak väčší prehľad o reálnom nasadení aplikácie a vedia, aké možné situácie môžu nastať pri jej používaní. Tieto rozdiely sa prejavujú už v efektívnosti testov na úrovni

jednotiek (anglicky unit testing), kde môže byť značný rozdiel medzi testom pripraveným vývojárom a testom od testera. Toto odlišné myslenie môže predstavovať bariéru v efektívnej komunikácii vývojárov a testerov.

Iným problémom zaradeným do tejto vrstvy je tzv. personalizovanie kódu. Veľa vývojárov berie osobne fakt, že niekto nájde v ich kóde chybu. Často si chyby odmietajú uznať, hľadajú chybu práve v testerovi.

Z ľudského hľadiska na vzťahy určite negatívne vplýva aj to, že vývojári a testeri spolu komunikujú hlavne vtedy, keď tester vývojárovi oznamuje chybu. To je pre vývojára dosť nepríjemná situácia. Aj keď vývojár chybu objektívne uzná, podvedome sa mu osoba testera spája iba s nepríjemnými zážitkami. To môže viesť k vybudovaniu negatívneho vzťahu k testerovi a možným veľmi vážnym konfliktom.

Manažér, ktorý ignoruje túto „ľudskú“ vrstvu zdrojov konfliktu sa môže vo svojom tíme veľmi ľahko ocitnúť medzi dvoma nenávidiacimi sa skupinami. Preto sa treba o vzťah medzi týmito dvoma skupinami starať, zabezpečiť stretnutia tímu a komunikáciu aj mimo práce. Ak sa tieto dve skupiny spoznajú aj v iných situáciách ako práve pri oznamovaní chýb, budú si oveľa lepšie rozumieť aj v práci.

Ďalšou možnosťou je aj zorganizovanie špeciálneho kurzu o zvládaní konfliktných situácii pre členov tímu. Okrem toho, že si účastníci kurzu osvoja schopnosti, ktoré im pomôžu lepšie komunikovať a zvládať konfliktné situácie sa zároveň spoznajú aj z inej stránky ako pracovnej, čo určite zlepší ich vzájomné vzťahy.

Problémy komunikácie a personalizovania kódu by sa mohli dať riešiť aj vhodným párovaním vývojára s testerom, ktorý testuje jeho kód. Predpokladám, že starší a skúsenejší tester môže pri komunikácii chýb mladšiemu vývojárovi čerpať zo svojich skúseností a naopak, skúsenejší vývojár by už nemusel trpieť prílišným zosobňovaním kódu a môže prijať svoje chyby objektívne.

*Organizácia*

Ak politika a manažment organizácie nepodporujú svojich testerov, nedostáva sa im a ich práci primeraný rešpekt a táto sa stáva oveľa ťažšou akou by mohla byť v skutočnosti. Opačný príklad, keď by sa tento problém týkal vývojárov v praxi nenastáva. Práve naopak, možný konflikt medzi testermi a vývojármi vidíme hneď ako si uvedomíme, že oproti nedoceneným testerom sa vývojári často preceňujú. Oni sú predsa tí, ktorí vykonávajú v spoločnosti kreatívnu činnosť, oni tvoria produkt. Testeri potom vrhajú svoju energiu do snáh o priznanie statusu, o vyrovnanie sa s vývojármi namiesto do svojej skutočnej práce – do testovania.

V prípade, že manažment organizácie nekladie dôraz na kvalitu produktu a neposkytuje podporu svojim testerom, nebudú ich rešpektovať ani vývojári. Navyše, ak manažér nevenuje pozornosť vzťahom medzi vývojármi a testermi a tieto sa vyhrotia, tak sa môže ľahko stať, že testerova primárna úloha sa stane takmer nesplniteľnou.

Problém môže ešte viac zhoršiť nesprávna štruktúra spoločnosti. Ak je oddelenie testerov úplne izolované od ostatných častí organizácie a navyše fyzicky umiestnené ďaleko od vývojárov, bez možnosti osobnej komunikácie, stáva sa čas potrebný na vyriešenie problému oveľa dlhším ako je v skutočnosti nutné.

Vývojári a testeri by teda mali pracovať relatívne blízko seba, prípadne sa môžu vytvoriť zmiešané tímy. Aj keď sa tým nevylúčia možné konflikty, urobí sa významný krok smerom k ich zvládaniu.

V neposlednom rade je potrebné, aby manažéri organizácie jasným spôsobom komunikovali smerom navonok potrebu testovania ako časti vývojového procesu, ktorý vedie k spoločnému cieľu – kvalitnému produktu.

## Záver

To, že medzi ľuďmi v tíme dochádza ku konfliktom je fakt, ktorý musíme akceptovať. Zároveň však nesmieme tieto konflikty nechať neriadené, keďže to predstavuje príliš veľké riziko pre projekt a pre tím. Manažér, ktorý by konflikty v tíme odignoroval môže mať na svedomí rozpad tímu, ktorý by pri inom riadení mohol dosahovať veľmi dobré výsledky.

V tejto práci manažér nájde prehľad o konfliktoch v tímoch, príčinách ich vzniku a možnostiach ich riadenia. O tom, že konflikty môžu v tíme vznikať na viacerých úrovniach sa môže presvedčiť na uvedenom príklade softvérového tímu a konfliktu medzi vývojármi a testermi.

## Použitá literatúra

1. Cohen C. et al.: Managing conflict in software testing. *Communications of The ACM*, Vol. 47, No. 1 (2004) 76-81.
2. Mackey K.: Stages of team development. *IEEE Software,* Vol. 16, No. 4 (1999) 90-91
3. Philips C: Team conflict: *How to Manage It*. Dostupné na http://chumans.com/skills3.htm (December 2005)

## Annotation

*Managing conflicts within a team*

Conflicts are a natural part of our lives. Nevertheless, it is necessary to manage the development of conflicts, because if left unmanaged conflicts can have a severely disrupting effect on a project. This paper describes many possible sources of conflicts in teams along with clues for managers to resolve them. On the example of a software development team and its typical type of conflict between software developers and testers it explains the role of managers in resolving conflicts in teams.

## Zhrnutie

Dôležitosť a zložitosť softvéru v súčasnom svete viedla k vzniku Softvérového inžinierstva. Skutočnosť, že náklady na tvorbu a údržbu softvéru sú netriviálne vedie mnohé spoločnosti k formalizácii a riadeniu interných procesov tvorby softvéru. Vzhľadom na povahu softvéru nie je možné ho „vyrábať“ štandardnými postupmi, ktoré sa používajú napr. na výrobu automobilov alebo stavbu domov. Kreatívna povaha tvorby softvéru je na jednej strane krásna a motivujúca pre jeho tvorcov no na druhej strane spôsobuje aj viaceré problémy s odhadovaním a meraním kvality softvéru ako aj s optimalizáciou jeho tvorby.

Softvérové inžinierstvo a v rámci neho manažment v softvérovom inžinierstve sa pokúšajú riešiť uvedené problémy aplikovaním systematických prístupov a istou mierou štandardizácie jednotlivých procesov.

Jedným z možných riešení je definovanie a dodržiavanie stratégie spoločnosti v oblasti podnikovej kultúry a formalizácie procesov pomocou zamerania sa na ich kvalitu. Esej s témou *Manažment kvality a vplyv na výsledok projektu* opisuje viaceré normy určené na zabezpečenie kvality – štandardy ISO a modely CMM. Myslíme si, že podobné snahy môžu mať značne pozitívny vplyv na celkovú kvalitu softvéru, ktorá by mohla byť ešte väčšia v prípade, ak by bolo možné skúsenosti a technológie jednotlivých spoločností zdieľať.

Nemenej dôležitými aspektmi pri tvorbe softvéru sú však aj samotní ľudia ako hlavný kapitál softvérovej spoločnosti a zákazníci ako odberatelia softvéru. Metódy riadenia vzťahov so zákazníkom ako aj spôsoby motivácie a výberu zamestnancov opisujú eseje *Manažment vzťahov so zákazníkmi* a *Zlepšovanie produktivity softvérových tímov*.

Pre úspech softvérového projektu však nestačí, aby mala spoločnosť formálne definované procesy. Každý softvérový projekt si vyžaduje aj projektového manažéra, ktorý uvedené procesy aj prostredníctvom členov tímu pretaví do praxe. Významnými procesmi sú najmä manažment rizík a odhadovanie a plánovanie projektu, ktoré v ideálnom prípade umožňujú projektovému manažérovi s dostatočnou presnosťou povedať v akom stave sa projekt nachádza a kam smeruje. Vďaka týmto informáciám je projektový manažér schopný včas identifikovať možné problémy a prijať nápravné opatrenia. Systematické prístupy k analýze rizík a metódy odhadovania opisujú eseje *Manažment rizík v softvérovom projekte* a *Odhadovanie v softvérových projektoch*.

Prijatie vhodných formálnych procesov spoločnosti a pochopenie ich významu členmi softvérového tímu umožňuje nielen tvorbu kvalitného produktu ale aj zlepšenie celkovej atmosféry v tíme, ktorý sa môže nielen spoľahnúť na svojho projektového manažéra ale aj pracovať s relatívnou istotou, že projekt bude úspešný. Toto všetko samozrejme platí len vtedy, keď sú jednotlivé procesy dobré, dobre definované a dodržiavané.

Na úrovni zloženia tímov a spolupráce ich členov je dôležité venovať dostatočnú pozornosť ich komunikácii a riešeniu konfliktov, ktoré môžu počas existencie softvérového tímu nastať. Problematikou týchto oblastí sa zaoberajú eseje *Vývoj tímu v*

*softvérovom projekte a vplyv na manažment*, *Komunikácia v tíme* a *Manažment konfliktov v tíme*.

Cieľom riadenia v tejto oblasti je vytvorenie zohraných softvérových tímov, ktoré dokážu spolu efektívne pracovať a zdieľať informácie. Veľkú pozornosť treba venovať práve riadeniu komunikácie v tímoch, pretože neefektívna komunikácia môže veľmi negatívne ovplyvniť produktivitu tímu a ohroziť tak úspešnosť projektu. Veľmi podobná situácie môže nastať v prípade neriešenia pretrvávajúcich konfliktov v tíme, ktoré môžu nastať napr. kvôli nepochopeniu rolí jednotlivých členov tímu alebo problémom s komunikáciou.

Keď zhrnieme hlavné myšlienky jednotlivých esejí a zamyslíme sa nad aktuálnym stavom v softvérovom inžinierstve ako aj tendenciou jeho vývoja do budúcnosti, môžeme povedať, že jedným z jeho cieľom je aj neustále zvyšovanie kvality softvéru a produktivity softvérových tímov. Tento cieľ sa snaží dosiahnuť pomocou riadenia procesov tvorby softvéru, ktorými sa zaoberá manažment v softvérovom inžinierstve. Myslíme si, že aj napriek niektorým pretrvávajúcim problémom sa mu tento cieľ darí úspešne napĺňať.

Michal Tvarožek (editor)

Manažment v softvérovom inžinierstve
Zbierka esejí

Prvé vydanie
Vydané FIIT
Slovenská technická univerzita
Bratislava

Náklad: 1 výtlačok
103 strán
2006