MANAŽMENT PRACOVNÝCH SÍL V SOFTVÉROVOM PROJEKTE

Skúsenejší vývojári, kvalitnejší produkt. Ale je to naozaj pravda?

Michal Jesenský

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

Abstrakt. *Súčasťou plánovania softvérového projektu je zostavenie dobrého tímu a rozdelenie úloh členom tímu. Na kvalite finálneho produktu sa podieľajú rovnakou mierou osobnostné vlastnosti členov tímu, ako aj ich skúsenosti a schopnosti. Pre manažérov nie je možné v mnohých prípadoch zostaviť tím iba z ľudí špecializujúcich sa v problémovej oblasti softvérového projektu, a preto musia využiť zamestnancov, ktorí nemajú takéto skúsenosti. Týmto vzniká otázka, či je potrebné za každú cenu vyhľadávať ľudí, ktorí sa v danej oblasti vyznajú, skombinovať expertov s ľuďmi, čo sa dokážu rýchlo učiť a prispôsobiť, alebo vybrať na projekt iba ľudí, ktorí sú flexibilní. Esej sa zaoberá vplyvom skúseností členov tímu na celkový tímový výkon a od toho odvodený vplyv schopností tímu ako celku na kvalitu finálneho softvérového produktu. Okrem toho popisuje fungovanie distribuovaného projektu ako celku a faktory, ktoré sa musia brať do úvahy a sú špecifické pre distribuovaný vývoj. Základné skúsenosti však musia získať členovia projektových tímov počas vzdelávania a v týchto prípadoch simuluje reálne prostredie a podmienky inštruktor tímu.*

Kľúčové slová: *pracovná sila, vývojové schopnosti, metriky kvality softvéru, distribuovaný projekt, inštruktor*

Úvod

Pri tvorení plánov softvérového projektu sa musia manažéri zaoberať hlavne otázkou nasadenia vhodných ľudí na riešenie projektu. Na výber vhodných ľudí pre projekt vplývajú najmä ich schopnosti, skúsenosti a povahové vlastnosti. Členovia tímu musia mať určité psychologické vlastnosti, aby tím pracoval ako celok a predišlo sa nezhodám a iným problémom fungovania tímu. Tím zostavený z ľudí, ktorí dokážu veľmi dobre spolupracovať, však vôbec nezaručuje úspech projektu, ak členovia nemajú dostatočné skúsenosti, ale aj v tomto prípade existuje riešenie, pretože najdôležitejšia schopnosť človeka je schopnosť učiť sa alebo prispôsobiť sa. Ak projektový manažér nedokáže zohnať ľudí, ktorí sa vyznajú v danej oblasti, musí zabezpečiť školenie pre, tých čo sa v nej nevyznajú. Od schopností členov tímu je odvodená schopnosť celého tímu a od nej kvalita finálneho produktu. Tento postup však nefunguje úplne rovnako v distribuovaných projektoch, pretože v nich spolupracuje niekoľko izolovaných tímov. Pre projekt ako celok je preto dôležité rozdelenie na čo najmenšie časti a zabezpečenie aktívnej komunikácie medzi tímami a centrálnym (riadiacim) tímom. Pretože v príklade, ktorý som využil spolupracovali tímy študentov z rôznych fakúlt, rozhodol som sa analyzovať možné úlohy pedagogických vedúcich, ktorí tieto tímy viedli.

Ohodnotenie schopností ľudí

Pri dosadzovaní úloh v softvérovom projekte je dôležité poznať členov tímu. Základnými faktormi, ktoré ovplyvňujú rozdelenie rolí a priebežné prideľovanie úloh, sú povahové vlastnosti členov tímu, skúsenosti s danou problematikou a schopnosti založené na inteligencii a zručnosti členov tímu. Podrobnejšie sa budem zaoberať ohodnotením schopností vývojového tímu na základe schopností a skúseností jednotlivých členov. Toto ohodnotenie vychádza z predpokladu, že schopnosti úzko súvisia so skúsenosťami.

Predpokladom je, že viac vzdelaný a skúsený tím vyprodukuje produkt vyššej kvality. Konštrukčná časť životného cyklu softvéru sa skladá z troch fáz, z ktorých každá vyžaduje rôzny typ zručnosti alebo skúsenosti na úspešné ukončenie. Fáza požiadaviek je zameraná na organizačné a medziľudské schopnosti. Zahŕňa interakciu so zákazníkmi, zisťovanie potrieb zákazníkov a organizovanie týchto potrieb do sady požiadaviek. Fáza návrhu si vyžaduje schopnosť tvoriť softvérové riešenia, ktoré spĺňajú identifikované požiadavky, a schopnosti vo vyhodnotení možných prístupov a nástrojov na implementáciu týchto riešení. Implementácia vyžaduje pracovné znalosti a zručnosť v špecifických vývojových nástrojoch, vrátane jazykov, operačných systémov, knižníc a vývojových prostredí. Zatiaľ čo integrácia a testovanie sú potrebné časti vývojového cyklu, toto hodnotenie sa zaoberá schopnosťami a skúsenosťami v produkčných fázach vývoja [1]. Podobné ohodnotenie schopností tímu sa dá aplikovať aj na tieto fázy s pravdepodobnosťou veľmi podobných výsledkov.

Každá fáza životného cyklu vyžaduje veľké množstvo rôznych schopností. Prístup opísaný v [1] k meraniu schopností vývojového tímu vyhodnocuje skúsenosti a schopnosti práve v zmysle týchto troch fáz životného cyklu. Napríklad tím veľmi schopný v návrhu, ale s malou zručnosťou v oblasti vývoja požiadaviek, môže vyprodukovať fantastický produkt, ale nemusí nevyhnutne splniť požiadavky zákazníka.

Ohodnotenie členov je zlúčené do množín úrovní podľa prínosu člena celému tímu. Tento prístup je založený na NASA CMS (National Aeronautics and Space Administration's Competency management system). CMS je framework aplikovaný na úrovni veľkých podnikov na kvantifikovanie vedomostí, schopností a možností pracovnej sily v NASA a na inteligentné manažovanie jej alokácie do projektov, ktoré ju vyžadujú. CMS ustanovuje štyri úrovne ohodnotenia vedomostí, schopností a skúseností jednotlivca. Pre každú úroveň sú vytvorené kritériá, ktoré musí zamestnanec spĺňať, aby bol zaradený do danej úrovne. Úrovne schopností podľa CMS sú jednoducho rozdelené na úroveň 1 až úroveň 4, kde úroveň 4 predstavuje najvyššie ohodnotenie schopností v danej fáze.

CMS síce zvyšuje mieru abstrakcie pri kategorizovaní ľudí, ale pretože každý človek je dôsledne testovaný, predtým ako je zaradený do danej kategórie, pre všeobecný a prehľadný pohľad na kvalitu tímu stačí toto jednoduché zaradenie do 4 skupín. Pri tvorbe plánov môžu vedúci a manažér plánovania jednoduchšie rozdeľovať ľudskú silu a odhadovať potrebné časy na dokončenie jednotlivých úloh v určitých fázach podľa tohto ohodnotenia. Napríklad členov v úrovni 1 dosadia na úlohy, ktoré im umožnia získať viac skúseností a zlepšiť sa v danej problematike a naopak členov z úrovne 4 môžu dosadiť na vedúce miesta, alebo ako dočasný dozor a podporu pri vykonávaní úloh, na ktorých sa podieľajú viacerí členovia. Zároveň kombinácia viac a menej schopných členov môže okrem splnenia danej úlohy priniesť ľahšie plnenie ďalších úloh, vďaka zvýšeniu schopností členov nižších úrovní.

Zároveň pomocou tohto ohodnotenia je možné vytvoriť štatistiku vplyvu kvality vývojového tímu na kvalitu softvérového produktu a porovnať rozdiely lepšie ohodnotených tímov s horšie ohodnotenými.

Efekt schopností tímu na finálny produkt

Kvalita softvérového produktu sa môže merať rôznymi spôsobmi a podľa toho môže byť opísaná rôznymi atribútmi. ISO/IEC 9126 je medzinárodným štandardom pre kvalitu softvérového produktu, ktorý reprezentuje kvalitu dodaného softvérového produktu v zmysle šiestich hlavných charakteristík: funkcionalita, efektívnosť, spoľahlivosť, použiteľnosť, udržiavateľnosť a prenosnosť. Tento štandard je podľa [4] spracovaný do tabuľky 1.

Tab. 1. Metriky kvality softvérového produktu [1][4].

ISO/IEC 9126 metrika	Účel metriky	Popis metriky
Adekvátnosť návrhu (AN)	Ako adekvátne sú vybrané moduly návrhu?	Podiel počtu modulov návrhu spĺňajúcich podmienky adekvátnej funkcionality k počtu modulov návrhu.
Adekvátnosť implementácie (AI)	Ako adekvátne sú vybrané jednotky zdrojového kódu?	Podiel počtu jednotiek zdrojového kódu spĺňajúcich podmienky adekvátnej funkcionality k počtu jednotiek zdrojového kódu.

Úplnosť funkčnej implementácie (UFI)	Do akej miery je úplná funkčná implementácia?	Jedna mínus podiel počtu funkcií zaznamenaných ako chýbajúce počas fázy vyhodnotenia funkcionality k počtu funkcií opísaných v požiadavkách.
Pokrytie funkčnej implementácie (PFI)	Ako správna je funkčná implementácia?	Jedna mínus podiel počtu funkcií zaznamenaných ako chýbajúce alebo nesprávne implementované počas fázy vyhodnotenia funkcionality k počtu funkcií opísaných v požiadavkách.
Nestálosť funkčnej špecifikácie (NFS)	Ako premenlivá je funkčná špecifikácia po definovaní základov projektu (project baseline).	Podiel počtu požiadaviek zmenených po zadefinovaní projektu k počtu požiadaviek.

Tabuľka 1 zobrazuje metriky použité v zachytení kvality softvérového produktu pre časť Vhodnosti štandardu ISO/IEC 9126. Tieto merania zachytávajú úroveň kvality softvéru z perspektívy funkčného chodu systému. Jediným rozdielom medzi metrikami prezentovanými v tabuľke a ISO/IEC 9126 je to, že kritérium funkčnej adekvátnosti bolo rozšírené, aby zahŕňalo moduly návrhu a jednotky zdrojového kódu.

Podľa úrovní schopností z predošlej kapitoly a metrík z tabuľky 1 boli pomocou ohodnotenia úspešnosti členov a projektov vyrátané korelačné koeficienty ukazujúce vplyv skúseností a schopností vývojového tímu na kvalitu softvérového produktu. Tieto koeficienty sú zachytené v tabuľke 2.

Tab. 2. Koeficient korelácie schopností ku kvalite softvéru [1][4].

Úroveň schopností	ISO/IEC 9126				
	AN	AI	UFI	PFI	NFS
Pož. 1	-0.415	-0.755	-0.022	-0.146	-0.062
Pož. 2	-0.419	-0.619	-0.042	0.038	0.190
Pož. 3	0.434	0.752	-0.013	-0.030	-0.094
Pož. 4	0.515	0.645	0.266	0.411	-0.212
Diz. 1	0.045	-0.314	0.246	0.166	0.052
Diz. 2	-0.513	-0.592	-0.302	-0.404	0.158
Diz. 3	0.433	0.693	0.059	0.219	-0.152
Diz. 4	0.413	0.358	0.288	0.375	-0.235
Impl. 1	N/A	-0.377	0.258	0.129	0.085
Impl. 2		-0.758	-0.406	-0.296	0.168
Impl. 3		0.367	-0.017	-0.094	-0.058
Impl. 4		0.597	0.307	0.375	-0.252

Pre adekvátnosť softvérového produktu z tabuľky jasne vyplýva, že zvýšenie počtu skúsenejších ľudí pozitívne koreluje s kvalitou produktu. Naopak ľudia z tried 1 a 2 majú negatívnu koreláciu a medzi koreláciami týchto tried a tried 3 a 4 je výrazný rozdiel (miestami väčší ako 1). Z týchto vzťahov vyplýva, že zvyšovanie počtu menej skúsených ľudí v projekte má silný dopad na zníženie kvality finálneho produktu v zmysle adekvátnosti.

Pre úplnosť funkčnej implementácie a pokrytie funkčnej implementácie je vidno, že iba vo fáze dizajnu majú triedy 3 a 4 kladnú hodnotu. V ostatných fázach, iba zvyšovanie počtu najschopnejších ľudí pozitívne ovplyvní kvalitu produktu. Vo fáze dizajnu a implementácie pozitívne korelujú aj úrovne 1, čo môže práve príčinou ďalej opísaných dôvodov.

Vplyv na nestálosť funkčnej špecifikácie bol presne opačný. Všeobecný predpoklad, že ľudia z vyšších tried budú presnejšie špecifikovať funkcionalitu a menej meniť túto špecifikáciu, sa potvrdil. Napriek tomuto správaniu je rozsah koeficientov korelácie prekvapivo nízky. Tým pádom predpoklad, že nestálosť sa prudko zníži so zvyšovaním schopností tímu, nie je celkovo naplnený. Naopak nízky rozsah koeficientov korelácie ukazuje, že schopnosti majú len nepatrný dopad na zníženie nestálosti špecifikácie. V [1] sa upozorňuje na tento fakt a navrhuje sa hlbší prieskum vplyvu vývojového procesu alebo zložitosti produktu na určenie dôležitejšieho faktoru, ktorý ovplyvňuje nestálosť funkčnej špecifikácie.

V niektorých prípadoch je možné pozorovať nižšiu hodnotu koeficientu korelácie pre úroveň 2 ako pre úroveň 1. Je to zaujímavý úkaz, pretože ľudia z úrovne 2 sú schopnejší, preto by mali pozitívnejšie vplývať na kvalitu produktu. Tento úkaz môže byť zapríčinený napríklad tým, že ľuďom s úrovňou 1 sú prideľované jednoduché úlohy, určené skôr na zaškolenie týchto členov, alebo úlohy nevyžadujúce si podstatné vedomosti v nejakej oblasti, a naopak ľuďom s úrovňou schopností 2 sú prideľované príliš ťažké úlohy, pretože sú chápaní, že majú skúsenosti, ale nie je brané do úvahy, že tieto skúsenosti môžu byť nedostatočné. Potom títo ľudia nesplnia svoju úlohu v predpokladanej kvalite.

Všeobecne platí, že ak chceme mať kvalitný produkt, musí ho vytvoriť skúsený tím. Tento výskum potvrdil viac-menej zrejmé tvrdenie, ale ukázal podrobnejší náhľad do situácie. Tento prieskum bol robený na niekoľkých centralizovaných tímoch, ale pri vývoji softvéru v distribuovanom projekte nastáva úplne iná situácia, na iné veci sa musí brať ohľad a kvalita tímov sa nerovná kvalite produktu, ak nie sú tímy dobre koordinované.

Manažment v distribuovaných tímoch

Globálne distribuovaný vývoj softvéru je zložitý. Takýto vývoj sa využíva najmä pri vývoji veľmi veľkých softvérových produktov, ktoré budú pracovať na globálnej úrovni a slúžiť používateľom z rôznych častí sveta. Základným problémom distribuovaného vývoja je, že komunikácia, koordinácia a riadiace mechanizmy, ktoré prirodzene fungujú v centralizovaných tímoch, nie sú dostupné alebo nie moc efektívne v distribuovaných tímoch. Udržať rovnaké pochopenie návrhu systému medzi distribuovanými tímami oddelenými fyzickými, časovými a kultúrnymi bariérami sa tým pádom stáva náročné. V týchto podmienkach naberá rola architekta o dosť väčší význam. Architekt musí prekonať tieto bariéry vytvorením zdieľanej úrovne pochopenia kontextu systému a jeho

problémovej domény. Zároveň manažovanie závislostí tímov si vyžaduje včasné rozhodovanie. Ak architekt nesplní dostatočne tieto požiadavky, môže to viesť k nefungujúcim tímom a neúspešným projektom.

Siemens Corporate Research (SCR) realizoval výskum s názvom Global Studio Project (GSP) zameraný na organizáciu a manažment distribuovaných tímov študentov z rôznym univerzít. V tomto projekte mali tímy vytvoriť systém s názvom MSLite na kontrolu vybavenia budov ako kúrenie, ventilácia, prístupy... Každé zariadenie malo svoj systém na kontrolu, preto bol použitý simulátor budiacej sústavy (Field System Simulator - FSS) na vytvorenie počiatočných konfigurácií týchto systémov [3].

Pre distribuovanie vývoja systému MSLite študentským tímom, jeho architektúra musela byť jasne členená, pretože architektúra rozhoduje, aká bude štruktúra systému, čo vplýva na organizáciu a manažment vývoja. Systém MSLite bol rozdelený na moduly, podľa ktorých bola rozdelená práca tímov. Samozrejme tieto moduly museli byť podrobne popísané, a ak bolo možné, rozdelené na pod-moduly, aby bolo jasné, na ktorej časti funkcionality má ktorý tím pracovať. Na pochopenie dynamiky systému bol použitý model komponentov a konektorov, v ktorom komponenty predstavovali nezávisle nasaditeľné jednotky funkcionality zložené z jedného alebo viacerých modulov a konektory predstavovali mechanizmy komunikácie medzi nimi. Dokopy pohľady na systém pomocou modulov a komponentov umožnili analýzu závislostí použitú na určenie úrovne koordinácie a kolaborácie potrebnej medzi tímami pracujúcimi na rôznych aspektoch systému. Závislosť je v tomto prípade definovaná ako ľubovoľný aspekt komponentu, ktorý závisí na inom komponente alebo mu niečo poskytuje (dátová štruktúra, typ správania...). Zároveň však museli byť brané do úvahy ostatné faktory ako časovanie a závislosť na zdrojoch medzi komponentmi na získanie kompletného a presného zobrazenia systému.

Po definovaní závislostí a ich pochopení a jednoznačnom prekonzultovaní, mohli byť softvérové moduly pridelené tímom na vývoj. Možnou organizačnou štruktúrou, ktorá sa často používa pri riadení distribuovaných projektov, je takzvaná „hub-spoke“ štruktúra. „Hub“ predstavuje centrálny tím zložený z hlavných rolí: produktový manažér, projektový manažér, manažér dodávateľov, manažér infraštruktúry, manažér požiadaviek, architekt a integračný inžinier. Každý „spoke“ je napojený na „hub“ a predstavuje jeden tím skladajúci sa z rolí ako vedúci tímu, architekt, manažér infraštruktúry, manažér zdrojov a vývojár [3].

Ako je vidieť z opisu GSP, vznikajú určité záležitosti, ktoré robia vývoj softvéru v takomto distribuovanom prostredí značne náročné. Po prvé, tvorba architektonických artefaktov a ich komunikácia medzi tímami si vyžaduje veľké úsilie. Požiadavky stakeholderov musia byť získané a pretransformované do jazyka problémovej domény, ktorý je zrozumiteľný pre komunikáciu medzi vývojovými tímami. Za druhé, tímy musia dobre porozumieť závislostiam medzi modulmi systému, aby ich vyvíjali účinne a efektívne. Medzi tímami a zákazníkmi musí byť realizované množstvo komunikácie a dohadovania, aby došlo k vzájomnému porozumeniu. Za tretie, pridelenie práce distribuovaným tímom musí byť dôkladne vytvorené berúc do úvahy párovanie jednotiek softvéru, stabilitu ich opisu a jasnosť, s ktorou sa o nich dá komunikovať medzi tímami [3].

Projektoví manažéri sa často snažia znížiť komunikáciu medzi tímami pridelením relatívne nezávislých modulov rôznym tímom. Tento prístup je však príliš náročný kvôli

existencii závislostí medzi modulmi, ktoré vyžadujú komunikáciu a kolaboráciu tímov. Jedným z kľúčových aspektov tímovej komunikácie kritickej pre úspešnosť distribuovaného projektu je zdieľané porozumenie navrhovaného systému [3]. Tento aspekt nie je dôležitý iba v počiatočnej fáze a vo fáze spracovania projektu, keď sa spracúvajú požiadavky na systém, ale je taktiež kritickým v neskorších fázach, pretože zachováva uvedomenie si princípov návrhu systému. Toto je veľmi dôležité, pretože architektúry smerujú k odchýleniam od pôvodnej špecifikácie, ak sú ich základné princípy zabudnuté alebo ignorované. Toto sa môže ľahko stať, ak je architektúra definovaná veľkým množstvom dokumentov.

V [3] je definovaný prístup rozdelenia práce distribuovaným tímom cez zdieľaný kontext projektu (Shared Project Context – SPC). V prístupe založenom na SPC je vytvorený model artefaktov a závislostí medzi nimi. Artefakty v SPC sú bežné aj pre fázu požiadaviek, aj pre fázu návrhu a architektúry. Slúžia na koordináciu vytvárania a synchronizáciu spracovania ostatných požiadaviek a architektonických artefaktov. Architekt by mal viesť tvorbu tohto kontextu, pretože jej kvalita výrazne ovplyvní kvalitu všetkých neskorších fáz projektu a zvýši efektivitu skorších fáz (v prípade vodopádového vývoja). SPC slúži najmä na jednotné pochopenie projektu tímami a zabránenie tvorbe viacerých realít (každý tím má vlastné pochopenie projektu) ukotvením základných konceptov problému a návrhu jeho riešenia.

Tvorba SPC najviac zaťažuje analytikov, návrhárov a architektov, ale pretože SPC poskytuje všeobecný a jednotný pohľad na problematiku, uľahčuje manažment distribuovaných tímov.

Myslím, že ak by SPC bolo skombinované s distribuovaným vývojom systémov ako napríklad SOA (Service Oriented Architecture), bolo by veľmi jednoduché plánovať a rozdeliť prácu medzi jednotlivé tímy pri zachovaní celkového obrazu vyvíjaného softvéru. Zároveň pri tvorbe jednotlivých služieb by bolo ľahšie rozdeliť prácu v rámci tímu a použiť monitorovanie kvality projektu ako v prípade centralizovaného tímu, pretože SOA predpisuje nezávislosť služieb.

V prípade GSP boli využité tímy zložené z vysokoškolských študentov a koordinované zamestnancami SIEMENS-u v centrálnom tíme. Tieto tímy mali pravdepodobne svojich vedúcich z pedagogického zboru fakúlt a tí zastávali niektoré roly, iné ako možné roly členov tímu. Takýto prístup môže priniesť veľmi veľa participujúcim študentom v projekte.

Roly inštruktora v riadení študentských projektových tímoch

Na stredných školách a univerzitách sa študenti učia ako pracovať v tíme na riešení softvérových projektov. Študenti si rozdelia dôležité úlohy medzi sebou podľa ich schopností, skúseností, času a povahových vlastností. Každý študentský tím má prideleného vedúceho z pedagogického zboru. Okrem dohliadania na priebeh projektu môže tento vedúci zastávať rôzne roly vyskytujúce sa v reálnych projektoch. Medzi tieto roly patria podľa [2] zákazník, architekt, tímový manažér a učiteľ.

Ako zákazník, ktorý zadáva študentom softvérový projekt, inštruktor môže poskytovať slovný alebo písomný opis projektu. Študenti musia vykonávať rozhovory o definovaní požiadaviek a pripravovať formálne špecifikácie na prebratie so zákazníkom

o prípadných úpravách. Inštruktor môže pôsobiť ako zákazník počas akceptačných testov projektu, aby zabezpečil, že tím postupuje správnym smerom počas každej iterácie projektu. Inštruktor zastávajúci rolu zákazníka môže tiež pôsobiť ako sprostredkovateľ komunikácie v prípade, že projekt bol zadaný externým zadávateľom.

Ako architekt špecifikuje inštruktor celkovú štruktúru riešenia a tým môže pomôcť študentom vybrať si správnu softvérovú architektúru. Toto je veľmi dôležitý výber, ktorý musí byť urobený dostatočne skoro, lebo ovplyvňuje všetok beh projektu. Neskoré zmeny architektúry môžu byť extrémne drahé a môžu výrazne ohroziť ukončenie projektu načas. Väčšinou nemajú študenti dosť skúseností vo výbere architektúry a vykonaní architektonických rozhodnutí, a preto je kľúčové, aby im inštruktor poskytol podporu počas tohto dôležitého procesu.

Ako tímový manažér zúčastnený v manažmente rizík, inštruktor musí dávať pozor, aby úsilie tímu nebolo ohrozené kvôli možnému nedostatku jemných zručností (komunikácia s okolím, komunikatívnosť, štýl práce...). Zároveň musí dohliadnuť na to, aby tieto zručnosti boli vytvorené a rozvíjané. Napriek tomu, že je zodpovednosťou študentov vytvoriť všetky artefakty vznikajúce z projektu, inštruktor môže pomôcť tímu držať sa rozvrhu poskytujúc spätnú väzbu založenú na revízii všetkých vyprodukovaných artefaktov. V tomto ohľade sa roly tímového manažéra a zákazníka môžu prekrývať, pretože rovnaká osoba poskytuje spätnú väzbu tímu z technického pohľadu na vec (revízia artefaktov), ako aj z pohľadu zákazníka (verifikácia implementácie produktu po každej iterácii).

Ako učiteľ musí inštruktor poskytovať vedenie každému tímu a individuálnym študentom sledovaním ich špecifických potrieb a odpovedaním otázok špecifickým pre ich projekty a týkajúcich sa zodpovedností v projekte. Okrem toho môže inštruktor poskytovať rady individuálnym študentom ohľadom riešenia konfliktov v tíme. Takáto rada poskytnutá v správnom čase môže zachrániť projekt pred katastrofou, pretože zabráni narastaniu nezhôd medzi členmi tímu a z toho ústiacej neschopnosti spolupracovať.

Podľa vlastnej skúsenosti a podľa vychádzania z faktu, že v školách sa študenti hlavne učia, väčšina študentov nemá, alebo má iba čiastočné skúsenosti s prácou v tíme od začiatku definovania požiadaviek až po nasadenie finálnej verzie softvéru. V týchto prípadoch je úloha inštruktora veľmi dôležitá, ale často študenti na neho neberú ohľad a pracujú podľa seba ako uznajú za najvhodnejšie. Dôležité je, aby inštruktor dokázal zhodnotiť schopnosti a skúsenosti tímu podľa toho vedel motivovať a viesť študentov reprezentovaním všetkých alebo niektorých rolí. Okrem toho môže študentom pomôcť a poradiť pri všetkých etapách vývoja softvéru a pri rozdeľovaní času potrebného na realizáciu úloh súvisiacich s projektom.

Záver

Manažment pracovných síl je dôležitou súčasťou pri vývoji softvérového produktu. Správne odhadnúť schopnosti, skúsenosti a povahové vlastnosti členov tímu prináša zníženie práce pri plánovaní a rozdeľovaní úloh. Všeobecne je zrejmé, že zvyšovanie počtu skúsených členov tímu sa pozitívne odzrkadlí na kvalite softvérového produktu, ale

prieskum ukázal, že nie vždy je to pravda. Samozrejme výsledná kvalita nezávisí iba od počtu skúsených ľudí, ale týmto som sa v eseji nezaoberal.

Kvalita softvéru vytváraného viacerými distribuovanými tímami závisí hlavne od kvality práce architekta a zvoleného prístupu ku komunikácii a kolaborácii tímov. Podrobnejšie túto problematiku skúmala firma SIEMENS.

Na to, aby mohli ľudia, hlavne mladí, podávať vysoký výkon práce v softvérových tímoch, musia sa to naučiť najlepšie na prípade čo najpodobnejšiemu realite. Tu naberá vysokú dôležitosť úloha inštruktora (vedúceho), ktorý dokáže tímy riadiť a simulovať externé roly, ktoré vystupujú v reálnych projektoch.

Použitá literatúra

1. Beaver, M.J., Schiavone, G.A.: The Effects of Development Team Skill on Software Product Quality. *ACM SIGSOFT Software Engineering Notes*, Vol. 31, No. 3 (2006) 1-5.
2. Kurkovsky, S.: Instructor Roles in Software Engineering Course Projects. Dostupné na internete: http://www.sersc.org/journals/IJAST/vol1/papers/09.pdf [cit: 2009 október]
3. Sangwan, S.R., Ros J.: Architecture Leadership and Management in Globally Distributed Software Development. *International Conference on Software Engineering*, Leipzig (2008), 17-22.
4. Vinod, V., Dhanalakshmi, J., Sahadev, S.: Software Team Skills on Software Product Quality. *Asian Journal of Technology*, Vol. 8, No. 1 (2009) 8-13.

Annotation

Workforce Management in Software Project

The assembly of a good team and the allocation of tasks to team members is a part of planning in software project. Personalities of team members as much as their experience and skills participate with equal measure on the quality of final product. In most cases, it is not available for managers to assembly team only from people specializing in given problem area, and so they have to use employees, who do not have these skills. Hereby a question appears, whether it is necessary at all cost to search for people, who understand given problem area, or combine experts with people who are able learners, or choose only flexible people to work on the project. The essay is about influence of team members' skill on the whole team production and the quality of final product derived from it. It also describes the core concepts of distributed software project and the factors, which have to be deliberated and are specific for distributed development. The members of project teams however, have to obtain the basic experience while learning at schools. In these cases, instructors simulate the real world conditions and environment.