Zlepšenie produktivity práce softvérového tímu

MARTIN HINKA

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava

Abstrakt. V posledných niekoľkých desiatkach rokov došlo k rozsiahlemu rozvoju informačných technológií. Zatiaľ čo výkonnosť hardvéru a kapacita sietí urobila pôsobivo dlhé kroky pomocou automatizácie výroby a technologickou inováciou, vývoj softvéru sa nezlepšil na rádovo rovnakej úrovni. Následkom toho softvérové komponenty informačných systémov chronicky zapríčiňujú meškanie projektov, prekročenie nákladov a nespokojnosť zákazníkov.

Je všeobecne uznávané, že zlepšenie produktivity vývoja softvéru požaduje rovnomerný prístup k trojici pilierov softvérového manažmentu: technológia, ľudia, proces.

Existuje niekoľko prístupov koordinácie problémov pri vývoji softvérových systémov. Opíšeme tri súčasné prístupy, ktoré sa používajú: veľký tresk, častá integrácia a periodická synchronizácia, a závadami-poháňaný prístup. Prístupy sa líšia v rozdielnom načasovaní a intenzite koordinácie. Taktiež sa budeme venovať faktorom, ktoré ovplyvňujú intenzitu koordinácie. Tieto sme zoradili do štyroch skupín: projekt, tím, systém, technológia.

Ďalej si opíšeme efekt učenia sa tímu a povieme si niečo o fenoméne mýtického človeko-mesiaca, softvérových charakteristikách systému, technológiách a nástrojoch.

Na záver si spomenieme niečo o ekonomike a manažmente počas vývoji softvérových systémov.

Úvod

Ako môžeme zlepšiť produktivitu práce ľudí a tímov? Pozrieme sa do minulosti a poučíme sa z chýb, znalostí vedomostí, ktoré sme tam získali. A tiež tak, že pozrieme do budúcnosti a budeme sa snažiť pomocou týchto skúseností vyhnúť chybám, ktoré by sme mohli urobiť [3].

Manažment v softvérovom inžinierstve, máj 2005, s. 1-12.

Pokiaľ sa všetci budú snažiť pracovať tak, aby sme si mohli v budúcnosti povedať, že pracovali najlepšie ako vedeli a najefektívnejšie ako sa dalo bude zlepšenie produktivity práce prirodzene zabezpečené. Zatiaľ máme pred sebou ešte dlhú cestu, pretože neustále je čo zlepšovať a dokonalá práca neexistuje. Môže byť dokonalá, resp. optimálna, len vzhľadom na niekoľko málo požiadaviek. Vzhľadom na všetky požiadavky nebude vždy dokonalá, pretože to vyplýva priamo z protichodnosti niektorých požiadaviek.

Informačné technológie a vývoj softvéru

Počas niekoľkých posledných desaťročí nastal prudký a rozsiahly rozvoj informačných technológií (IT). Tento rozvoj pomohol organizáciám dosiahnuť mnoho pracovných aj strategických úspechov.

Nielen organizácie, ale aj bežní spotrebitelia majú úžitok z IT, pretože redukuje faktory trhu spôsobené geografickou separáciou, cenovú neprehľadnosť a oneskorenie informácií.

Ľudia si uvedomili že informácie majú dnes čím ďalej vyššiu cenu, a preto sa snažia ich mať čím viac v čo najkratšom čase. Oneskorený príchod informácií už môže byť zbytočný, pretože informácie už nemusia byť aktuálne

Z toho vyplývajúce zvýšenie dopytu po IT produktoch a službách tvorí nové výzvy pre dodávanie IT riešení umocnených vývojom hardvéru, softvéru a sieťových komponentov a to pri neustále sa znižujúcich časoch na vývoj.

Zatiaľ čo rýchlosť hardvéru a kapacita sietí urobila pôsobivo dlhé kroky pomocou automatizácie výroby a technologickou inováciou, vývoj softvéru sa nezlepšil na rádovo rovnakej úrovni.

Následkom toho softvérové komponenty informačných systémov chronicky zapríčiňujú meškanie projektov, prekročenie nákladov a nespokojnosť zákazníkov.

Je tu niekoľko faktorov, ktoré spôsobujú obtiažny vývoj softvéru, zvlášť zložitosť softvéru, ktorá požaduje zásah človeka počas jeho tvorby. Následkom toho nie je možné realizovať plne automatizovaný vývoj softvéru. Napríklad použitím automatického generovania kódu.

Okrem toho inovačná povaha softvéru spôsobuje jeho ťažké podvolenie sa skúsenostiam a vedomostiam tímu počas projektov.

Nakoniec - nehmatateľnosť softvéru - komplikuje merania a kvantitatívne analýzy, ktoré sú nutné pre neustále sa zlepšujúcu produktivitu.

Piliere softvérového manažmentu

Je všeobecne uznávané, že zlepšenie produktivity vývoja softvéru požaduje rovnomerný prístup k trojici pilierov softvérového manažmentu: technológia, ľudia, proces [1].

Technológia

Zjemneniu stavebných technológií softvéru bolo venované veľké úsilie s významnými výsledkami. Napríklad, sofistikované kompilátory a skriptovacie technológie zvýšili rýchlosť programovania. Robustné pomocné nástroje dovolili jednoduché vyhľadávanie a odstraňovanie chýb, ladenie systému, jednoduché nastavenie a konfiguráciu.

Komunikačné aplikácie a vytváranie sietí umožnili udržať projekt a systémové informácie prehľadné a rozčlenené, čo je pri obrovských množstvách informácií jeden z veľmi dôležitých faktov. Schopnosť vedieť udržať prehľad a rozdelenie medzi kvantami informácií tak, aby sa v nich vedelo orientovať veľké množstvo ľudí je vzácna, ale dá sa nadobudnúť praxou. Používanie dobrých nástrojov tomu prirodzenou cestou napomáha. Je dôležité, aby sa informácie dali členiť takmer prirodzene zo svojej podstaty, pretože umelé, násilné členenie vyvoláva pocit dezorientácie a vnáša neprehľadnosť.

Ľudia

Ľudia sú dnes nutnou a neodmysliteľnou súčasťou pri získavaní informácií a pri vývoji projektov. Kvôli človeku - smerujúcej povahe vývoja softvéru, prínos z technologických zlepšení nemôže byť plne realizovaný bez spôsobilých zamestnancov. Základom je teda získať, alebo si vychovať spoľahlivých a schopných zamestnancov.

Avšak, investovanie do ľudských zdrojov vyžaduje dlhodobé plánovanie a záväzok no nevytvára okamžitú odmenu. Existuje však CMU / SEI People Capability Maturity (P-CMM) rámcový systém, ktorý poskytol odporúčania na to, ako sa majú uskutočňovať organizačné zmeny tak, aby umožňovali lepší manažment a vývoj zamestnancov.

Proces

Tretím spôsobom ako dosiahnuť stúpanie produktivity je zjemnenie a zušľachtenie samotného vývojového procesu. Prínosy z tohto spôsobu zlepšenia procesu nie sú limitované iba obmedzením sa na zrýchľujúcu sa prácu na vývoji, ale taktiež znižujú úsilie na veľmi náročné a drahé opravné aktivity. Bez správneho vývojového procesu, projektový tím môže operovať iba chaotickým spôsobom, ktorý má následok nízku produktivitu a slabú kvalitu systému.

Procesné modely a politiky

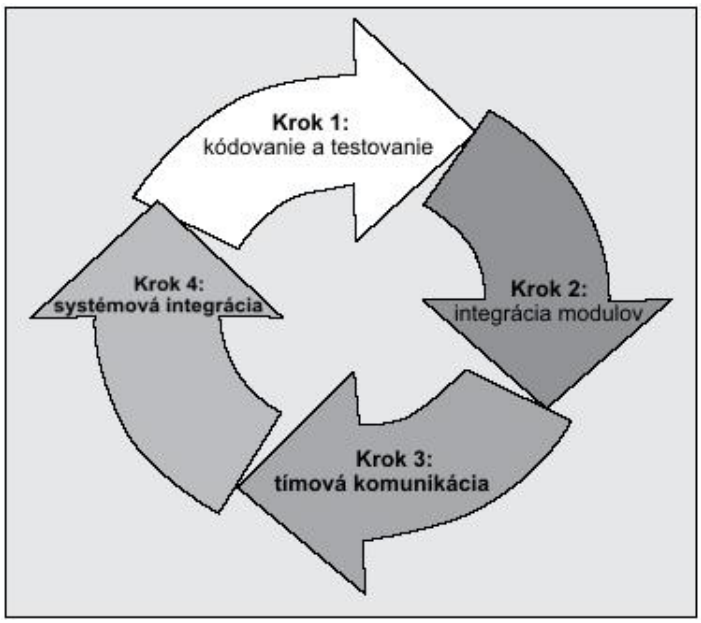
Literatúra z minulých rokov ako napríklad *Brooksov inkrementálny vývoj* a *Boehmov špirálový model* predpokladá, že vývoj systému by mal byť radšej skôr akt pozvoľného zlepšovania ako násilné spájanie softvérových súčiastok. Brooks objavil že inkrementálny vývoj, v ktorom sa prelína práca na vývoji s obdobím testovania

a hľadania chýb, vedie k jednoduchému spätnému sledovaniu a prirodzenému prototypovaniu.

Rámcové systémy pre inkrementálny vývoj, aj pokiaľ sú kvalitné, často neponúkajú presný postup ako inkrementálny vývoj môže byť optimálne implementovaný. Zvlášť v prípadoch, keď zoberieme do úvahy široké spektrum systémov, zamestnancov a technických faktorov.

Prikročíme teda k množstvu kvantitatívnych procesných modelov a politík na zlepšenie organizácie inkrementálneho vývoja. Hlavná ťarcha spočíva na fáze *konštrukcie systému*, ktorá je stupňom projektu v ktorom sa uskutočňuje kódovanie, čiže implementácia systému. Špecifické činnosti, ktoré sa vyskytujú v priebehu konštrukcie systému sú *kódovanie a testovanie*, *integrácia modulov* (resp. integrácia modulov so zvyškom systému), *tímová komunikácia* (napríklad príklady, príručky, náhľady, stretnutia k projektom atď.) a *systémová integrácia* (systémové oživovanie).

Tieto činnosti sú zobrazené na Obr. 1 a sú sformované do *konštrukčného cyklu.* Na konci každého cyklu sa hotová časť systému (pozostávajúca z kolekcie stabilných softvérových modulov) stane základom, na ktorom môže byť založený ďalší vývoj resp. ďalšia činnosť. Typicky je potrebných niekoľko konštrukčných cyklov, kým je implementovaný a otestovaný celý systém [1].



Obr. 1. Činnosti počas konštrukčného cyklu.

S výnimkou kroku 1, činnosti zobrazené na Obr. 1 sú vzájomne koordinované. Väčšina konštrukcií reálne používaných systémov na svete vyžaduje koordináciu medzi projektovými účastníkmi, akými sú vývojári, testeri, návrhári, a systémový používatelia.

V tomto kontexte vývoja softvéru, Kraut a Streeter popísali koordináciu, zamerajúc sa na činnosti, ktoré umožňujú prácu rozdielnych ľudí na projekte majúcom presnú definíciu a presnú špecifikáciu, čiže presne vieme čo sa ide vytvárať. Delenie a vzájomné poskytovanie si informácií a súlad úsilia týchto ľudí medzi sebou, čiže koordinácia ľudí, informácií a procesu. Teda koordinácia vývoja softvérového systému a problémov s tým spojených.

Koordinácia problémov a jednotlivé prístupy

V tejto časti si vysvetlíme rôzne prístupy koordinácie a rozhodnutí v skutočných softvérových organizáciách. Špecificky si prediskutujeme otázku: *"Kedy je najlepší čas na koordináciu?"*

Správne zodpovedať na túto otázku je veľmi dôležité, pretože oneskorenie načasovania koordinácie za bod určitý bod, alebo príliš malá koordinácia, vedie k drahému prepracovaniu. Súčasne, predčasná, alebo prehnaná koordinácia, môže byť kontraproduktívna, pretože môže narušiť prácu na vývoji.

Opíšeme tri prístupy, ktoré sa v súčasnosti používajú: veľký tresk (Big-Bang), častá integrácia a periodická synchronizácia (frequent integration and periodic synchronization), a závadami-poháňaný (fault-driven) prístup. Líšia sa rozdielnym načasovaním spustením koordinácie [1].

Big-Bang

Big-Bang je prístup, kde celá koordinácia nastáva na konci projektu. V tomto prístupe sú kroky 2, 3, a 4 z Obr. 1 pozdržané až pokiaľ nie je kompletný krok 1. Tento prístup nasleduje *vodopádový model* a tak samo o sebe nejde o inkrementálnu vývojovú techniku.

Od tohto času koordinácia nie je aktívne organizovaná. tento prístup má mimoriadne nízke nároky na organizáciu projektu a preto je veľmi vhodný pre malé a schopné tímy pracujúce na dobre definovanom projekte. Pri tomto je schopnosť učenia sa tímu nižšia ako pri inkrementálnom modely [2].

Hlavným nedostatkom Bing-Bang koordinácie je rozšíriteľnosť. Softvérové súčiastky v systéme sa môžu vzájomne ovplyvňovať. Ak nie je tento nedostatok odstránený včas, tak následne sa stáva ťažko opraviteľný a zapríčiňuje ďalšie nedostatky, ktoré sa môžu vyskytnúť neskôr v projekte. Takéto vedľajšie efekty robia Bing-Bang prístup veľmi drahým na použitie pre väčšie projekty [1].

Častá integrácia a periodická synchronizácia

Softvérové organizácie teraz uskutočňujú integráciu modulov (tak ako v kroku 2 na Obr. 1) oveľa častejšie - často denne. Je to dobre dokumentovaný a publikovaný prístup adoptovaný firmou Microsoft. Ide o tzv. "*Daily Build and Smoke Test*" a bol použitý pri vývoji veľkého množstva projektov v mnohých firmách [1]. Mimo častej integrácie modulov, pravidelnej tímovej komunikácie a systémovej integrácie ktoré sú vykonávané zabezpečuje kvalitu produktu už jeho samotná konštrukcia.

Zatiaľ čo častejšia koordinácia na modulovej a systémovej úrovni pomáha zmierňovať efekt degradácie, dôležitá otázka zostáva nezodpovedaná: *"Ako dlho má trvať vývoj v každom cykle?"*

Všimli sme si, že časovanie koordinácie často prebieha spôsobom *ad hoc* v mnohých organizáciách. V dôsledku toho, by sa však tieto organizácie mali riadiť aspoň základnou koordinačnou politikou, kde koordinácia je intenzívnejšia na začiatku projektu, uvoľnuje sa počas stredu, a stáva sa zase intenzívnou blízko konca projektu. Takáto koordinačná politika môže byť vysvetlená ako výsledok dvoch faktorov: *tímového učenia sa a systémovej stability* [1].

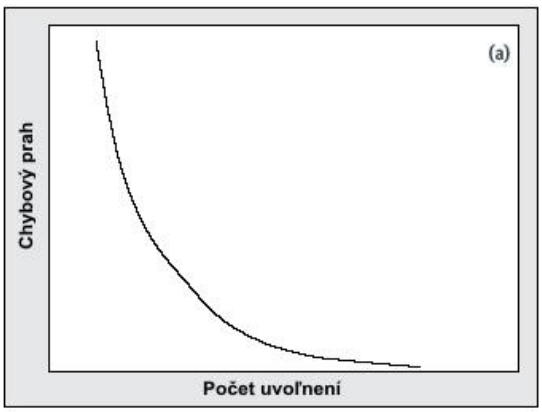
Závadami-poháňá koordinácia

Zatiaľ čo časovo-založené koordinačné politiky majú jasné organizačné prínosy, nejavia sa plne schopnými pokryť dynamiku vyvíjajúceho sa projektu.

Pri závadami-poháňanej koordinačnej politike špecificky platí, že koordinácia je naliehavejšia, keď sa zdá že systém nie je synchronizovaný, inak je vhodné v práci na vývoji pokračovať. V dôsledku toho koordinačné rozhodnutia by mali byť nejako zviazané k aktuálnemu stavu systému [2].

S pokročilým vývojom a nástrojmi riadenia projektu, je teraz možné získať údaje o chybách systému (*system fault data*) a ďalšie príbuzné metriky pomerne v skorých fázach projektu. Použitím aktuálnej početnosti týchto chýb a prísnych metrík, projektoví manažéri môžu plánovať koordináciu vždy, keď priemerné náklady na opravu chýb presahujú očakávané hodnoty resp. začnú stúpať.

Obr. 2 ukazuje krivku chybových prahov, ktorá môže byť odvodená s týmto zdôvodnením. Tvar prahovej krivky závisí na špecifických faktoroch projektu akými sú napríklad, zložitosť systému, veľkosť a skúsenosti tímu, vývojové prostredie, doména resp. oblasť projektu, a čas dosiahnuteľný pri konštrukcií systému.

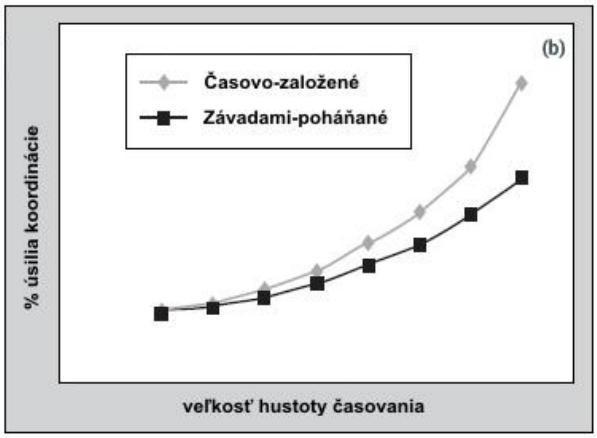


Obr. 2. Závadami-poháňaná koordinácia

(a) krivka chybových prahov (fault treshold curve)

Pri závadami-poháňanej politike, koordinácia nastáva pri uvoľnení modulu, keď počet chýb presiahne prahovú hodnotu stanovenú pre toto uvoľnenie. Zostupný svah krivky zaisťuje, že na začiatku je síce veľké množstvo chýb, ale po uvoľneniach niekoľkých modulov tento počet klesá. Koordinácia nemôže nastať až kým sa neuvoľní niekoľko málo modulov. Naproti tomu, prahy sa znižujú pri veľkom počte uvoľnení, čo je povzbudením pre koordináciu, lebo práca na vývoji má extrémne vysokú kvalitu.

Obr. 3 porovnáva časovo-založenú (*timed-based*) a závadami-poháňanú (*fault-driven*) koordinačnú politiku a ukazuje, že závadami-poháňaná koordinácia je často efektívnejšia. Použitie údajov o chýb systému pre vytvorenie koordinačných rozhodnutí je z čiastočne priaznivé pre projekty bez nahusteného časovania (čo nastáva, keď sa požaduje vysoký pomer práce na dosiahnutý čas) [1].



Obr. 3. Závadami-poháňaná koordinácia

(b) zisk zo závadami-poháňanej koordinácia (benefit of fault-driven coordination)

Faktory ovplyvňujúce koordináciu

Konceptuálne, intenzita koordinácie je pomer úsilia použitého na koordináciu v jednom konštrukčnom cykle. Zoradili sme teda faktory ovplyvňujúce intenzitu koordinácie do štyroch skupín: projekt, tím, systém, technológia [1].

Projekt

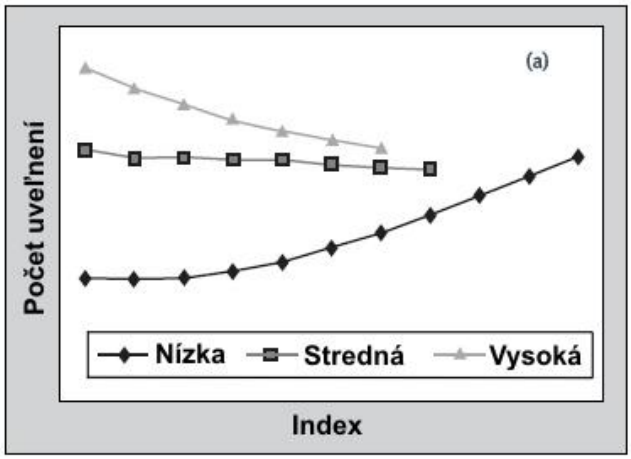
Kritickým faktorom ovplyvňujúcim produktivitu tímu je dovolený čas vývoja systému. Pre danú množinu požiadaviek, môžeme dosiahnuť kratší konštrukčný čas pri väčšom tíme a teda pri drahšej koordinácií medzi členmi tímu. Na druhej strane, zvyšovanie veľkosti tímu zmenšuje produktivitu na jedného člena.

Tento fenomén mýtického človeko-mesiaca (mythical man-month phenomenon) naznačuje, že pridávaním členov do tímu, za určitý bod môžeme skutočne predĺžiť trvanie projektu. Liek na konfrontáciu s veľkým tímom je zaviesť hierarchickú komunikačnú štruktúru a tak sa relatívne malé skupiny vývojárov môžu sústrediť na dobre prepojiteľné systémové komponenty.

Skúsenosti tímu a efekt učenia sa

Krivka na Obr. 4 predpokladá, že projektové tímy, ktoré môžu stabilizovať moduly veľmi rýchlo si môžu dovoliť menej častú koordináciu na projekte už skôr. Naproti tomu s narastajúcou voľnosťou, je potrebná častejšia koordinácia pokiaľ nie je dosiahnutá dostatočná spoľahlivosť a stabilita. Napríklad projekt uskutočnený menej skúseným tímom môže dosiahnuť stabilitu oveľa pomalšie, takže tím musí byť koordinovaný oveľa častejšie.

Indikáciou pomalej stabilizácie je, že hlavné prvky systému (také ako systémové rozhrania a moduly riadiace moduly) pokračujú v podstupovaní zmien až do neskorých fáz projektu.



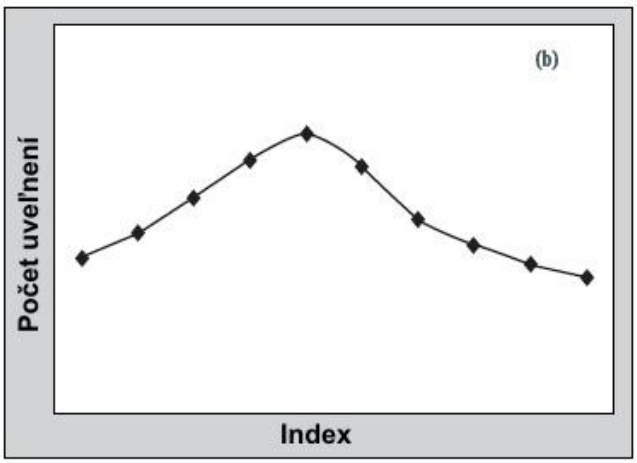
Obr. 4. Zmeny v krivke koordinačnej politiky

(a) efekt systémovej stabilizácie (effect of system stabilization rate)

Mnohé štúdie ukázali až desaťnásobnú priepasť v produktivite medzi nováčikom a zručným vývojárom. Jednou z čŕt prvotriednych vývojárov je ich schopnosť získavať projektovo-špecifické znalosti a vyhnúť sa tak väčšiemu prepracovaniu, dokonca aj v neznámej projektovej doméne resp. v neznámom projektovom prostredí.

Dôkazom tohto *"procesu učenia"* je, že vývojový tím stabilizuje novo vyvíjané moduly oveľa efektívnejšie ako predošlé a dochádza tak k pokroku schopností tímu a k rýchlejšiemu a kvalitatívne lepšiemu pokroku v projekte.

Zmeny v krivke koordinačnej politiky na Obr. 5 ukazujú, že pre takýto tím je vhodná relatívne úzka koordinácia na začiatku projektu, čo urýchľuje proces učenia. Tím by sa mal potom sústrediť na programovanie úloh s menšími zákrokmi takmer až do konca projektu, kedy je zase potrebná úzka koordinácia na zmenšenie rizík plánovania. Koordinačná schéma na Obr. 5 je odporúčaná pre skúsený tím pracujúci na projekte v novej doménovej oblasti.



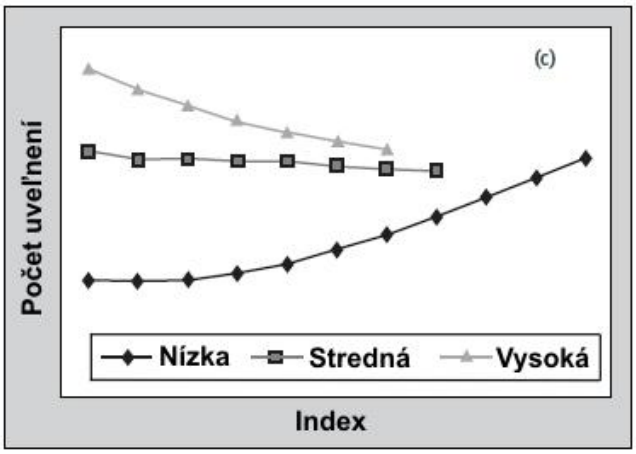
Obr. 5. Zmeny v krivke koordinačnej politiky

(b) efekt procesu učenia tímu(effect of team learning)

Softvérové charakteristiky systému

Čím vyššia je zložitosť systému, tým by mala byť intenzita koordinácie vyššia. Tento nárast však nie je v žiadnom prípade priamoúmerný. V zložitejších systémoch sa nesúlad v jednotlivých moduloch stáva rádovo ťažšie opraviteľný ako pri systémoch s menšou zložitosťou. Intenzívnejšia koordinácia preto pomáha udržať chybovosť pod kontrolou.

Obr. 6 ukazuje optimálnu krivku koordinácie pre rôzne úrovne systémovej zložitosti. Náklady na celkovú koordináciu taktiež narastajú so zložitosťou systému. Jeden z aspektov na Obr. 6 je, že nám poskytuje základ pre navrhovanú ekonomiku. Pokiaľ čelíte obmedzeným prostriedkom a zdrojom alebo neprávnemu načasovaniu projektu, tak projektový manažéri sú často v pokušení preskočiť návrh systému, aby sa viac venovalo produktívnej implementácií. Často by bolo pre nich lepšie investovať čas do prvotriedneho návrhu systému aby predišli náročnému a drahému "haseniu", resp. prerábaniu systému v priebehu konštrukcie.



Obr. 6. Zmeny v krivke koordinačnej politiky

(c) efekt zložitosti sytémov(effect of system complexity)

Technológie a nástroje

Zmeny v prínosoch možno merať tým ako ľahko členovia tímu vedia skĺbiť v rámci svojho pracovného prostredia vývoj a koordináciou. Ďalším dôležitým merítkom je podpora chápania projektu z danej doménovej oblasti. Inak povedané tím má presne vedieť, čo, kde, kedy a prípadne ako bude robiť. Preto podľa Obr. 6 by mala byť koordinácia intenzívnejšia, keď nastavajú efektívnejšie resp. väčšie zmeny projekte.

Napríklad v organizácií so sofistikovanou CASE podporou, je tím vedený ku koordinácií oveľa častejšie. Technologické inovácie ponúkané novými nástrojmi sa častokrát považujú za pomôcky umožňujúce zrýchlenie implementácie a celého vývoja projektu. V prípade správneho použitia, môžu taktiež znížiť nadbytočnú koordináciu.

Organizácie zaoberajúce sa vývojom softvéru sa teraz tešia zo širokého výberu nástrojov a technológií, od ktorých si sľubujú zvýšenie produktivity vývoja. Aj keď v priemysle, ktorý praktizuje tzv. "kreatívnu deštrukciu" sú rýchlosť a kvalita, teda výhody zo získaných technológií prenasledované vyššími očakávaniami používateľov.

Je preto dôležité vytvoriť dobrý a efektívny manažment projektového procesu, aby sme pomocou neho dokázali čo najlepšie využiť zamestnancov a technológie, ktoré máme k dispozícií.

Ekonomika vývoja projektu.

Jednou zo zaujímavých sfér budúceho výskumu je vývoj modelov a teórií rôznorodých softvérových systémov. V predchádzajúcich štúdiách sa na softvérových systém pozerá ako na súbor (viac či menej) identických modulov. Zatiaľ čo tento homogénny prístup

má jasné analytické výhody, je veľa situácií, kde softvérové moduly môžu obsahovať významné rozdiely vo veľkosti, zložitosti a funkcionalite.

Možnosťou je vytvoriť postupnosť, v ktorej sú moduly vyvíjané vzhľadom na obchodné faktory, ako napríklad funkcionalita, finančné plány a náklady na vývoj. Inak povedané, je potrebné vyvíjať ekonomickú základňu pre uvoľnenie softvérových modulov. Je teda dôležité založiť vývoj projektu na aktuálnych a premyslených zdrojoch ekonomických a neekonomických zdrojoch.
Mierkou pre zrelosť softvérových organizácií je či dokážu podávať konzistentnú, resp. rovnakú produktivitu počas dlhšieho obdobia. Veľa štúdií o produktivite sa zameriava na znižovanie priameho implementačného úsilia.

Produktivita tímu je výrazne ovplyvnená tým ako dobre je rozdelené koordinačné úsilie. Pokiaľ nie je koordinácia kontrolovaná, je málo pravdepodobné, že kvalita softvéru bude dosiahnutá rýchlejšie jednoduchým pridaním viacerých ľudí do projektu. Správny proces koordinácie a návrhu je preto kľúčom k lepšej produktivite a vyššej kvalite výsledného produktu.

Mnoho organizácií skôr nazerá na proces vývoja ako na zdržujúci ako na urýchľujúci produktivitu. Je to preto, že proces vývoja je často použitý cez donútenú postupnosť projektových úloh, bez explicitného zamyslenia sa nad koordinačnými aktivitami.

Slepo hľadať produktívne zlepšenia v snahe vyhovieť zákazníkovým očakávaniam je len sen, pokiaľ je len malá šanca ich splniť. Veľa projektových manažérov je neschopných spraviť citlivú analýzu pre štúdium následkov zmenených požiadaviek na projektové náklady, alebo plánovanie. Vlastnosti projektu ako systémová zložitosť, faktor stabilizácie a učenie sa tímu nám poskytujú smernice ako stanoviť dopad zmien projektu v plánovaní projektu [1].

Záver

Tento článok sa pokúša spojiť priepasť medzi analytickým a empirickým pohľadom na manažment projektu, aby poskytol smernice ako najlepšie riadiť softvérový projekt a pomohol tak zlepšiť produktivitu práce v softvérovom tíme.

Opisuje základné piliere softvérového manažmentu ako aj prístupy, ktoré sa dnes používajú pri koordinácií projektov.

Ďalej sa venuje faktorom ovplyvňujúcim koordináciu projektu, efektu učenia sa tímu, fenoménu mýtického človeko-mesiaca, softvérovým charakteristikám systému, technológiám a nástrojom používaným pri koordinácií a vývoji projektu.

Na záver je opísaná ekonomika a manažment projektu počas vývoja softvérových systémov.

Použitá literatúra

1. Robert Chiang, Vijay S. Mookerjee: Improving Software Team Productivity. In Communications of the ACM, Vol. 47, No. 5 (May 2004), 89-93.
2. Liu Xanfeng, Liu Guanguiu: Coordination in software develepment, review of "Improving Software Team Productivity, www presentation, (Februar 2005), 28 pages
3. Walt Scacchi: Uderstanding and improving software productivity, Institute for Software Research, University of California, Irvine USA, www presentation, (Februar 2004), 44 pages

Annotation

Improving software team productivity

Over the last few decades the widespread adoption of information technology (IT) was come. While hardware speed and network capacity have made impressive strides through manufacturing automation and technological innovation, software development has not improved under the same order of magnitude. As a result, the software component of information systems chronically causes project delays, cost overruns, and customer dissatisfaction.

It is widely recognized that improving software development productivity requires a balanced approach toward the three pillars of software management: technology, people, and process.

Many approaches of software development coordination problems exist. We describe three current approaches being used: Big Bang, frequent integration and periodic synchronization, and fault-driven. In each approach, there is different coordination trigger and coordination intensity. We describe the factors that have affect to coordination intensity. These have we clustered into four groups: project, team, system, and technology.

Next we describe the team learning effect and we said something about mythical man-month phenomenon, software characteristics, technologies and tools.

At the end we note something about software system development economics and management.