# Vplyv koordinácie procesu vývoja softvéru na produktivitu softvérového tímu

JOZEF WAGNER

*Slovenská technická univerzita*
*Fakulta informatiky a informačných technológií*
*Ilkovičova 3, 842 16 Bratislava*
wagner00@student.fiit.stuba.sk

**Abstrakt.** Už od počiatku bol softvérový priemysel v kríze. Softvér bol a stále je nespoľahlivý, dodávaný po stanovených termínoch, pružne nereagujúci na meniace sa potreby, neefektívny a drahý. Dokonca aj dnes, problémy so softvérovými systémami sú bežné a často publikované. Hlavným dôvodom je, že koordinácia aktivít sa stáva komplikovanou s rozsahom a narastajúcou zložitosťou projektu a preto sa problémy pochopiteľne častejšie vyskytujú pri vývojoch rozsiahlych softvérových systémov.

Preskúmame úlohu formálnej a neformálnej komunikácie pri koordinovaní práce na softvérových projektoch. Väčšina súčasných podporných nástrojov pre koordináciu používa procedúry formálnej komunikácie, ale ukazuje sa potreba zapojenia aj procedúr neformálnej komunikácie.

Existuje niekoľko prístupov koordinácie problémov pri vývoji softvérových systémov. Opíšeme tri súčasné prístupy, ktoré sa používajú: veľký tresk, častá integrácia a periodická synchronizácia, a chybami poháňaná koordinácia. Prístupy sa líšia v rozdielnom načasovaní a intenzite koordinácie. Taktiež sa budeme venovať faktorom, ktoré ovplyvňujú intenzitu koordinácie.

Záver eseje patrí motivácií ako ďalšiemu prostriedku na zvýšenie produktivity.

## Úvod

Ako je známe, už od počiatku bol softvérový priemysel v kríze. Softvér je nespoľahlivý, dodávaný po stanovených termínoch, pružne nereagujúci na meniace sa potreby, neefektívny a drahý, a bol počas predchádzajúcich tridsiatich rokov. Dokonca aj dnes, problémy so softvérovými systémami sú bežné a často sú publikované v prestížnych článkoch na internete.

Problémy sa pochopiteľne častejšie vyskytujú pri vývojoch rozsiahlych softvérových systémov, pretože koordinácia aktivít sa stáva komplikovanou

*Manažment v softvérovom inžinierstve, december 2005, s. 1-9.*

s rozsahom a narastajúcou zložitosťou projektu. Ich následky sa prejavujú vo forme meškania projektov, prekročenia nákladov a nespokojnosti zákazníkov.

V jednej ankete, ktorá ma zaujala, sa mali projektoví manažéri meškajúcich projektov vyjadriť, čo podľa ich názoru bola hlavná príčina, že nedodržali časový plán celého projektu. Až 50 % opýtaných respondentov uviedlo, že produktivita ich softvérového tímu nebola taká vysoká ako sa pôvodne očakávalo. Bolo to však naozaj spôsobené zníženou produktivitou softvérového tímu [1]?

## Koordinácia softvérového tímu

Koordinácia [2] bola zadefinovaná ako usmernenie vynaloženého úsilia jednotlivcov k dosiahnutiu spoločných, explicitne predložených cieľov a integrácia alebo vzájomné poprepájanie rozdielnych častí organizácie za účelom splnenia kolektívnej množiny úloh.

Koordinácia v procese softvérového vývoja znamená, že rozliční ľudia pracujú na spoločnom projekte, pričom si vymieňajú informácie a ich aktivity sa vzájomne preplietajú. Musia mať však rovnaký pohľad na vyvíjaný softvér najmä z hľadiska jeho budúcej funkcionality, organizácie a prepojenia s už existujúcimi softvérovými systémami. Aby ho postavili efektívne, musia zdieľať podrobnú špecifikáciu návrhu a informácie o vývoji jednotlivých softvérových modulov. Aby som to zhrnul, musia skoordinovať svoju prácu takým spôsobom, aby sa čo najrýchlejšie úspešne ukončila, nič sa nerobilo zbytočne a vo výsledku všetko do seba zapadalo.

## Charakteristika softvérového vývoja

Dosiahnutie vyhotovenia úspešného softvérového systému vyžaduje dôslednú koordináciu celého vynaloženého úsilia v cykle softvérového vývoja. A práve táto koordinácia sa nedosahuje vždy najľahšie.

Základnou črtou väčšiny softvérových systémov je práve ich rozsiahlosť, ktorá zabraňuje jednotlivcovi alebo malej skupine ich vytvorenie alebo dokonca ich dokonale pochopenie. Ak by softvérový systém bol malý, celý vývoj by mohol byť efektívne koordinovaný, pretože jednotlivec alebo malá skupina by mohli riadiť svoju prácu a zamerať sa na všetky detaily implementácie. Často rozsiahle projekty sú oveľa úspešnejšie, ak jedna, často výnimočná osoba so znalosťami z aplikačnej domény ako aj znalosťami softvérového riadenia koordinuje projekt. Avšak toto je nereálne pre rozsiahle softvérové projekty, kde veľkosť systému sa meria v miliónoch alebo desiatkach miliónoch riadkov kódu a životnosť projektu v rokoch.

Problémy koordinovania rozsiahlych projektov doplňuje miera neistoty. Pod neistotou môžeme rozumieť nepredvídateľnosť softvéru ako aj výsledkov úloh softvérových inžinierov. Na rozdiel od výroby, vývoj softvérového produktu nie je rutinná aktivita. Veľa softvérových systémov sú jedinečné projekty s neexistujúcimi prototypmi aplikácií alebo systémami, ktoré by sa jednoducho modifikovali alebo zmenili. S vývojom softvéru i miera neistoty narastá, pretože špecifikácia funkcionality

softvéru sa v čase mení, ako sa menia potreby používateľov a vyvíjajú konkurenčné systémy.

Veľký rozsah a neistota v softvéri by bol skoro zanedbateľný problém, keby softvérový produkt nevyžadoval precíznu integráciu zo svojich softvérových komponentov. Veľa softvérov je zložených z tisícok modulov, ktoré musia do seba zapadať, aby softvérový systém fungoval korektne. Slabá koordinácia medzi podskupinami produkujúcimi softvérové moduly môže viesť k chybám pri integrovaní samotných modulov.

V softvérom inžinierstve sa ukázalo, že praktické skúsenosti tímov z prechádzajúcich projektov nevyriešili problémy koordinácie vo veľkých softvérových projektoch.

Na potlačenie softvérovej krízy sa najčastejšie ponúkajú tri prístupy: (1) technické nástroje - od nových pracovných staníc k syntaxou riadením editorom až po vyššie programovacie jazyky, na zvýšenie produktivity jednotlivých vývojárov, (2) stavebnicovosť - technická ako napr. objektovo–orientované programovanie a manažérska, ako delenie do tímov podľa požiadaviek a (3) formálne procedúry - technické ako CASE nástroje, špecifikačné nástroje a manažérske ako testovacie plány, rozpisy odovzdaní a dokumenty požiadaviek.

Aj keď tieto techniky nepochybne prispeli k miernemu rastu v produktivite softvéru za posledných dvadsať rokov, zaoberali sa iba čiastočne problémami koordinácie vo vývoji softvéru. Nástroje na zvýšenie produktivity programátorov ako jednotlivcov nevyriešili koordinačné problémy. Podobne, vrstvové architektúry a štruktúrované programovacie techniky môžu zredukovať množstvo rozhraní medzi modulmi, avšak ľudia z rozdielnych tímov sa stále musia dohadovať, čo sa postaví a musia vhodne skladať moduly softvéru.

## Komunikácia ako prostriedok zlepšenia koordinácie

Popredné výskumy ukazujú, že formálna a neformálna komunikácia sa najlepšie hodí pre rôzne typy aktivít. Pod formálnou komunikáciou rozumieme písomnú komunikáciu alebo štandardné stretnutia. V prípade softvérového vývoja, formálna komunikácia zahŕňa techniky ako napr. písanie dokumentov špecifikácie, formálne špecifikačné jazyky a automatizované ohlasovanie a stopovanie chýb v programe. Tieto techniky kontrastujú s neformálnou komunikáciou, pod ktorou rozumieme osobnú a interaktívnu komunikáciu. Formálna komunikácia je užitočná pre koordinovanie transakcií v skupinách a organizáciách, avšak často zlyháva tvárou v tvár neistote, ktorá je typická pre väčšinu práce so softvérom. Za týchto podmienok môže byť neformálna komunikácia potrebná pre koordináciu.

Práve kvôli vzájomnej previazanosti jednotlivých skupín pracujúcich na vývoji softvérového projektu, neformálna, medziľudská komunikácia by mohla byť cenná metóda dosiahnutia potrebnej koordinácie. Ale v rozsiahlejších tímoch, neefektívnosť vzájomnej komunikácie každej dvojice ľudí môže zabrániť používaniu neformálnej komunikácie ako praktickej techniky na riešenie problémov koordinácie.

## Koordinácia problémov a prístupy

Mnoho softvérových organizácií po celom svete by mohli z veľkej časti eliminovať koordinačné problémy, keby si dokázali správne odpovedať na otázku: „Kedy je najlepší čas na koordináciu?“ Odpoveď na túto otázku je veľmi dôležitá, pretože zdržiavanie procesu koordinácie od určitého bodu môže viesť k drahému prepracovávaniu softvéru, avšak predčasná koordinácia môže byť kontraproduktívna, pretože môže narušovať prácu na vývoji softvéru.

V súčasnosti sa používajú tieto tri prístupy [3]:

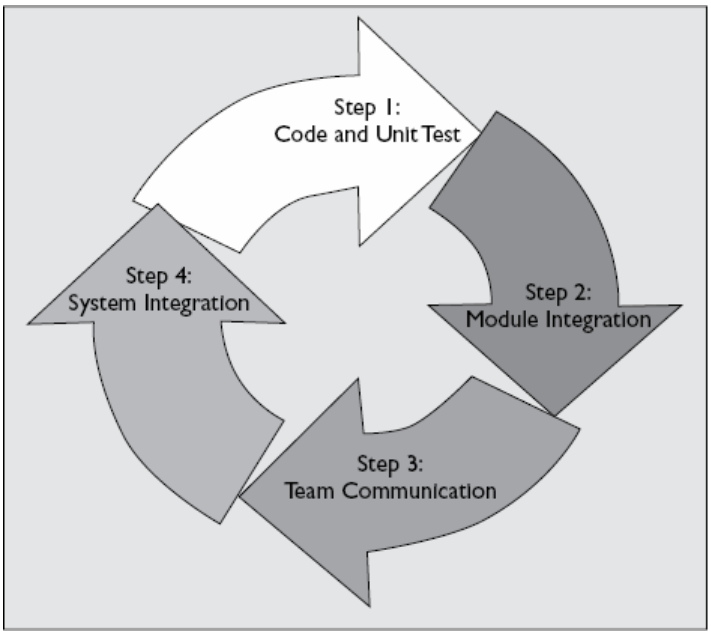
1. veľký tresk
2. častá integrácia a periodická synchronizácia
3. chybami poháňaná koordinácia

Tieto prístupy sa od seba odlišujú rozdielnym spustením procesu koordinácie.

### Veľký tresk (Big Bang)

Ide o 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 skompletizovaný krok 1. *Big Bang* nasleduje vodopádový model a tak nejde o inkrementálnu vývojovú techniku. Od načasovania koordinácie už sama o sebe nie je aktívne organizovaná, a preto má tento prístup mimoriadne nízke nároky na organizáciu projektu a je veľmi vhodný pre malé tímy pracujúce na dobre rozpracovanom projekte.

Hlavným nedostatkom tejto koordinácie je rozšíriteľnosť. Softvérové komponenty v systéme môžu mať vzájomne komplikované vzťahy. Ak sa tieto vzťahy nezjednodušia dostatočne skoro, projekt sa stane komplikovaným a horšie modifikovateľným v budúcnosti. Takéto vedľajšie efekty robia prístup *Bing Bang* veľmi nákladným pre väčšie projekty.




**Obr.1.** Činnosti počas konštrukčného cyklu. (zdroj [3] )

### Častá integrácia a periodická synchronizácia

Softvérové organizácie v tomto prístupe teraz uskutočňujú integráciu modulov (krok 2 v Obr. 1) oveľa častejšie, často denne. Ide o dobre zdokumentovaný a často publikovaný prístup prebraný softvérovým gigantom, firmou Microsoft. Takzvaný „*Daily Build and Smoke Test*" bol použitý pri vývojoch mnohých projektov v tejto firme. Popri častej integrácii modulov, prebieha komunikácia v tímoch a systémová integrácia, aby sa zabezpečila kvalita softvérového produktu počas celého vývoja.

Zatiaľ čo častá koordinácia na úrovni modulov aj systému pomáha zmierňovať vedľajšie efekty (ako napr. komplikované rozhrania modulov), dôležitá otázka zostáva nezodpovedaná. Ako dlho má trvať vývoj v každom cykle ?

V mnohých organizáciách si môžeme všimnúť, že základná koordinačná politika prebieha nasledovne. Koordinácia je najintenzívnejšia na začiatku projektu, počas neho poľavuje a blízko konca projektu znovu stúpa. Tento trend môže byť vysvetlený ako výsledok dvoch faktorov: tímového učenia sa a systémovej stability.

### Chybami poháňaná koordinácia

Zatiaľ čo, koordinačné politiky založené na čase majú jasné organizačné prínosy, nejavia sa plne schopnými pokryť dynamiku vyvíjajúceho sa projektu. Všeobecne platí, že koordinácia je oveľa naliehavejšia, keď sa zdá, že systém stráca synchronizáciu, inak je vhodné nechať prácu na vývoji softvéru pokračovať. Z tohto

vyplýva, že koordinačné rozhodnutia by mali byť nejako späté s aktuálnym stavom systému.

S pokročilým vývojom a nástrojmi riadenia projektu, je teraz možné získať údaje o chybách v systéme a ďalšie príbuzné metriky v skorých fázach projektu. Projektoví manažéri s použitím týchto údajov môžu plánovať koordináciu vždy, keď priemerné náklady na opravu chýb začnú rásť.

Tento typ koordinácie je mimoriadne prínosný pre zhustené časové plány, teda také, kde pomer požadovanej práce k dostupnému času je vysoký.

## Faktory ovplyvňujúce koordináciu

Intenzita koordinácie je pomer úsilia vynaloženého na koordináciu v jednom konštrukčnom cykle. Faktory ovplyvňujúce intenzitu koordinácie by sa dalo rozčleniť do štyroch skupín:

- projekt
- tím
- systém
- technológia

### Projekt

Kritickým faktorom ovplyvňujúcim produktivitu tímu je prípustný čas vývoja softvérového systému. Pre danú množinu požiadaviek, kratší čas vývoja softvéru sa dosiahne, ak bude na ňom pracovať väčší tím, avšak za cenu drahšej koordinácie medzi členmi tímu. Na druhej strane, zväčšovanie veľkosti tímu znižuje výslednú produktivitu na jedného člena.

Tento fenomén naznačuje, že neuvážené pridávanie ďalších členov do tímu môže od istej hranice skutočne predĺžiť trvanie projektu. Jediným rozumným východiskom pre veľké tímy je ich rozdelenie na relatívne menšie skupiny a zavedenie hierarchickej komunikačnej štruktúry. Takto sa tímy vývojárov môžu sústrediť na dobre prepojiteľné komponenty systému.

### Skúsenosť tímu a efekt učenia sa

Mnohé štúdie ukázali až desaťnásobný rozdiel v produktivite medzi nováčikom a skúseným vývojárom. Jenou z čŕt prvotriednych vývojárov je ich schopnosť získavať projektovo-špecifické vlastnosti a vyhnúť sa väčšiemu prepracovávaniu, dokonca v pre nich neznámej projektovej doméne. Dôkazom tohto procesu učenia je, že vývojový tím implementuje a optimalizuje novo vyvíjané moduly oveľa efektívnejšie ako predošlé a tak dochádza k zlepšeniu schopností tímu a k rýchlejšiemu a kvalitatívne lepšiemu pokroku v projekte.

### Softvérové charakteristiky systému

Čím väčšia je zložitosť systému, tým väčšia by mala byť intenzita koordinácie. Tento pomer však v žiadnom prípade nie je priamoúmerný. V zložitých systémoch sa každý nesúlad v jednotlivých moduloch ťažko opravuje a patrične zvyšuje náklady. Intenzívnejšia koordinácia preto pomáha udržiavať chybovosť v rozumnej miere. Náklady na celkovú koordináciu taktiež narastajú so zložitosťou systému.

Pokiaľ nie je na projekt vyčlenených dostatok finančných, materiálnych aj ľudských zdrojov alebo je projekt v časovom sklze, projektoví manažéri sú často v pokušení preskočiť fázu návrhu, aby sa viac času venovalo produktívnej implementácií. Často by pre nich bolo lepšie investovať ešte nejaký čas do prvotriedneho návrhu, aby predišli časovo náročnému a drahému prerábaniu systému v priebehu konštrukcie.

### Technológie a nástroje

Zmeny v produktivite možno merať tým ako ľahko členovia tímu vedia skĺbiť svoje pracovné prostredie s vývojom a koordináciou. Iným meradlom môžu byť skúsenosti z danej problémovej oblasti. Jednoducho povedané, ak tím presne vie, čo má robiť, ako, kedy a kde, produktivita bude vyššia, ako keby si najskôr naštudoval celú problémovú doménu.

V organizácii so sofistikovanou CASE podporou je tím vedený ku koordinácií oveľa častejšie. Technologické inovácie, ktoré ponúkajú nové nástroje, sa často krát považujú za pomôcky na zrýchlenie vývoja celého projektu. V prípade správneho použitia dokážu odbúrať nadbytočnú koordináciu. Organizácie zaoberajúce sa vývojom softvéru si v súčasnosti môžu vybrať z pestrej palety nástrojov a technológií.

Je preto úlohou dobrého a efektívneho manažmentu projektového procesu rozhodnúť sa, či finančné prostriedky radšej investuje do nových a kvalitných technológií, alebo do ďalších zamestnancov.

## Motivácia ako prostriedok zlepšenia produktivity

Osobne si myslím, že najlepší spôsob, ako zvýšiť produktivitu v ktorejkoľvek inštitúcií je vhodným spôsobom motivovať jej zamestnancov. To znamená vytvoriť im také pracovné prostredie, v ktorom sa budú cítiť príjemne a nebude ich pri práci nič vyrušovať. Predsa len, ak zamestnanec trávi jednu tretinu pracovného dňa na nejakom pracovisku, začne toto miesto nazývať druhý domov. Zároveň by mali byť za svoju prácu adekvátne finančne ohodnotení.

V mnohých softvérových firmách sa osvedčil prémiový spôsob odmeňovania zamestnancov. Pracujúci dostávajú primeraný mesačný plat a na konci roka sa im podľa zisku firmy a ich pracovnej činnosti vypočítavajú prémie. Tie tvoria u tých najúspešnejších zamestnancov niekoľko násobok ich mesačného platu. Prémie sa samozrejme rozdávajú iba, ak bola daná firma počas roku úspešná a zisková. Takto sú zamestnanci motivovaní a spoločnými silami sa snažia o to, aby podnik čo najlepšie prosperoval.

Ďalším výborným spôsobom zvýšenia produktivity zamestnancov je, podľa môjho názoru, umožniť im ďalej sa vzdelávať, rozvíjať svoje schopnosti, ale hlavne im umožniť úspešný postup v povolaní. Je zrejmé, že ak sa zamestnanec vo firme cíti dobre a vie, že to niekam môže dotiahnuť, pracuje usilovnejšie, aby si to jeho nadriadení všimli.

Na predchádzajúcich riadkoch som uviedol niekoľko spôsobov pozitívneho motivovania pracovníkov v podniku. Avšak motivácia ako prostriedok na zlepšenie produktivity môže byť aj negatívna.

Typickým príkladom negatívneho motivovania zamestnancov sú praktiky, ktoré prevádzkuje nielen manažment slovenskej verejnoprávnej televízie. Hlavnou myšlienkou tejto motivácie je navodiť u zamestnancov strach o stratu zamestnania. Slová personálneho manažmentu sú prosté: „Ak sa vám u nás nepáči, tak môžete odísť“. Neviete si predstaviť, aký obrovský účinok majú tieto slová najmä na starších pracovníkov tohto podniku, pretože vedia, že čím je človek starší, tým ťažie si nájde novú prácu. Personálny manažment toho patrične využíva a okrem množstva práce, ktorú stále kladie na plecia takýchto zamestnancov, nechce im zvyšovať plat. A tak sa pýtam, ako je možne, že vyštudovaný inžinier, ktorý pracoval tri desiatky rokov v tejto televízii, má sotva taký mesačný plat, ako nastupujúci ekonóm ?

## Záver

Správne načasovaná koordinácia celého procesu vývoja softvéru umožní firmám dodržiavať dohodnuté termíny. Zákazníci budú spokojní a firmy takto ušetrený čas a finančné prostriedky, ktoré by inak padli na „hasenie“ projektu, môžu investovať práve do motivácie svojich zamestnancov a nových technológií. Tým dosiahnu ešte väčší nárast produktivity a za ten istý čas dokážu zvládnuť a úspešne dokončiť viac projektov.

Celkom na záver by som si dovolil odpovedať na otázku položenú v úvode. Aký bol hlavný dôvod meškania projektov? Podľa môjho názoru išlo vo väčšine prípadov práve o pochybenie projektového manažmentu, pretože neodhadli korektne plán projektu a neskoordinovali správne celý proces vývoja softvéru.

## Použitá literatúra

1. Scacchi, W.: Understanding software productivity. *Advances in Software Engineering and Knowledge Engineering*, D. Hurley (ed.), Volume 4, (1995) 37-70.
2. Kraut, E.R., Streeter A.L.: Coordination in software development. *ACM Press*, Vol.38, No. 3 (March 1995) 69-81.
3. Chiang, R.I., Vijay S., Mookeriee.: Improving software team productivity. *ACM Press*, Vol.47, No. 5 (May2004) 69-81.

## Annotation

*Coordination's affect within software development process on the software team productivity*

Since its inception, the software industry has been in crisis. Software has been unreliable, delivered late, unresponsive to change, inefficient and expensive. Even today, problems with software systems are common and highly-publicized occurrences. Major contributor is the problem of coordinating activities while developing large software systems, therefore coordination becomes much more difficult as project size and complexity increases.

In particular, we examine the respective roles of formal and informal communication mechanism in coordinating work on software projects. Most of the existing coordination support tools have used formal communication procedures, but there is a need for informal communication procedures as well.

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 coordination. We describe factors that affect coordination intensity.

The end of essay belongs to motivation as another remedy for productivity increase.