# Potreba plánovania a adaptácie v softvérových procesoch

VILIAM GANZ

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

**Abstrakt.** Plánovanie vývojových procesov sa v súčasnej dobe dá považovať za nevyhnutnosť, minimálne pri rozsiahlejších projektoch. Bez dobrého plánu sa vývoj stáva nekoordinovaný a tým sa vo výraznej miere znižuje efektivita a zvyšuje časová náročnosť. Vytvoriť dobrý plán je však náročná záležitosť vyžadujúca skúsenosti a zručnosti. Ani takýto plán však nezaručuje úspešný priebeh projektu. Neočakávané a nepredvídateľné udalosti môžu značne ovplyvniť vývoj. Cieľom tohto dokumentu je poukázať na potrebu plánovania a adaptácie plánov podľa okolností, ktoré neboli očakávané.

## Úvod

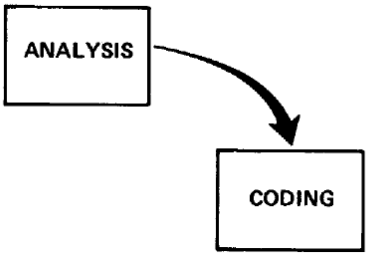
V súčasnej dobe sa počítače stávajú cenovo dostupnejšie aj pre širšie masy. Tým narastá potreba vývoja softvéru, keďže každý človek má inú predstavu o ideálnom softvéri. Navyše sú počítače v neustálej evolúcii, z čoho vyplýva potreba evolúcie softvérových systémov a programov. Dalo by sa povedať, že softvér je v neustálom kolobehu vývoja a adaptácie. Zložitejšie softvérové produkty však na svoj vývoj potrebujú aj viac programátorov a ich organizácia je zložitejšia. Jednoduchý program je schopný v rozumnom čase vytvoriť aj jeden programátor, ktorý si sám manažuje čas. Ak sa ale jedná o veľký produkt, je potreba vývojárov rádovo väčšia a tým vzniká potreba organizácie. Navyše pre zákazníka je časový aspekt vývoja veľmi dôležitý. Preto je potrebné odhadnúť trvanie samotného vývoja a, samozrejme, zabezpečiť kvalitu produktu. Ako teda vylepšiť softvérové procesy?

## Plánovanie

Čo je to vlastne proces? Proces sa podľa [3] zameriava na otázku „Ako?“, a produkt na otázku „Čo?“. Proces je formálne definovaný v IEEE podľa [3] ako sekvencia krokov vykonávaných z určitých dôvodov a v SEI ako Organizovanie ľudí, automatizovanej

*Manažment projektov softvérových a informačných systémov, október 2008, s. 1-8.*

podpory, procedúr a štandardov do pracovných aktivít na dosiahnutie určitého konečného výsledku. Podobne softvérový proces je definovaný v SEI podľa [3] ako množina aktivít, metód a transformácií ktoré používajú ľudia na vývoj a údržbu softvéru a previazaných produktov, napr.: plány produktov, kódy, testy a používateľské príručky. Tieto aktivity však nemôžu byt chaotické. Malé softvérové produkty sa skladajú z menšieho počtu aktivít, ako môžeme vidieť na obrázku 1. V takomto prípade často človek, ktorý robil analýzu, samotný projekt aj vyvíja a dodáva.



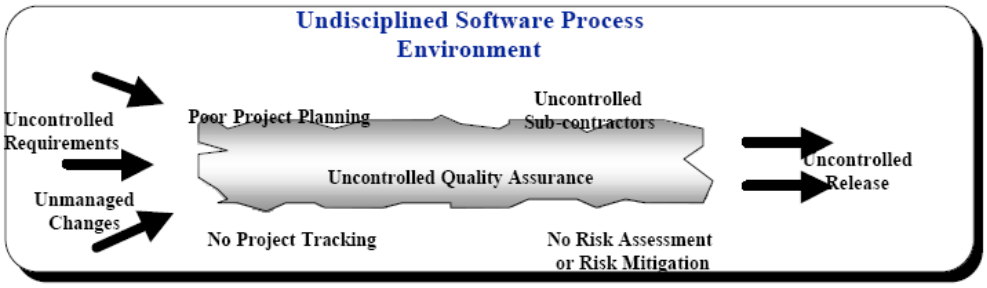

**Obr. 1.** Implementačné kroky v malých programoch podľa[2]

Vývoj väčších softvérových produktov sa skladá z viacerých krokov, ktoré nemôžu byť chaotické. V chaotických prostrediach sa často vyskytuje tzv. požiarnicky syndróm[3]. Programátor, ktorý vytvára bezchybný kód, ostáva nepovšimnutý a programátor, ktorý „hasí“ chyby v kritických momentoch, je hrdina. Čiže, v konečnom dôsledku, kto robí chyby a potom ich po sebe opravuje, je lepšie ohodnotený ako ten, ktorý vytvoril bezchybný kód v požadovanom čase. Na obrázku 2 môžeme vidieť graficky znázornené prostredie chaotického vývoja. Myslím si, že dobre zachytáva problematiku prostredia bez dobrého plánovania a jeho dodržiavania. Na začiatku máme nekontrolované požiadavky, ktoré bývajú často nereálne a zbytočné. Plánovanie je často riešené len v mysli vedúceho tímu, a tým znemožňuje kontrolu stavu vývoja a ani spätné ohodnotenie. Väčšinou takýmto spôsobom vznikne chybový produkt, ktorý je na poslednú chvíľu opravovaný, čo zvyčajne zahŕňa hrubé zásahy do pôvodných myšlienok programátorov. Nie zriedka sa na poslednú chvíľu vyskytnú požiadavky na zmenu funkcionality typu „ešte toto by sme chceli“ a ak nemáme obojstranne podpísanú špecifikáciu požiadaviek z prvej fázy vývoja, ľahko sa dostaneme do nepríjemných situácií, nehovoriac o nárazovej nočnej práci navyše, aby sa stihli termíny.

Typické chaotické prostredie podľa[3]:

1. výkon učený kompetenciou a odhodlaním pracujúcich ľudí
2. konzistencia a dodržanie štandardov je riadená manažmentom priorít – zvyčajne je najväčšou prioritou rozvrh
3. vysoká kvalita a výnimočný výkon je možný, len ak dokážeme zamestnať najlepších ľudí

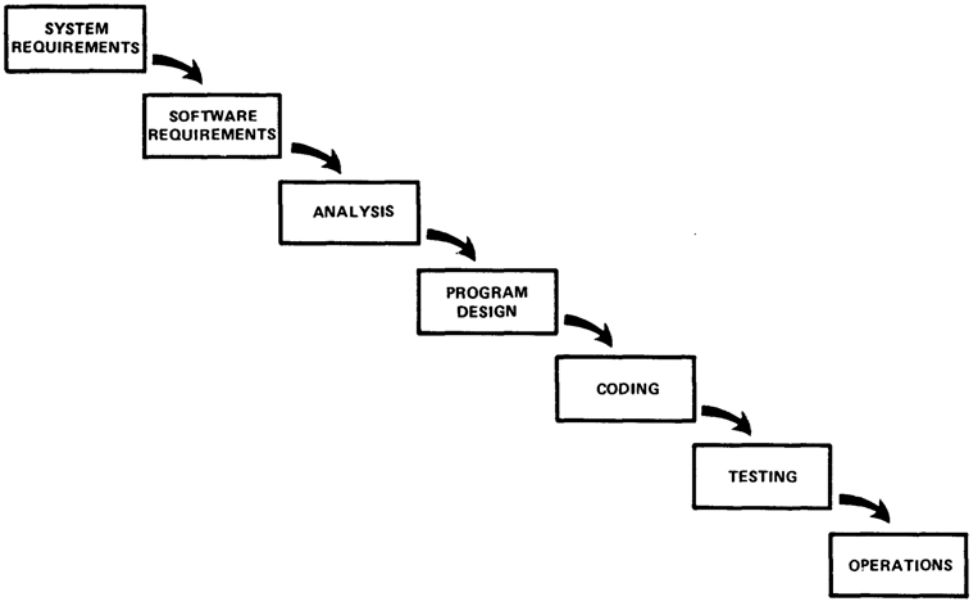
4. nepredvídateľnosť – dobrého a zlého – charakterizuje iniciálnu úroveň Organizácie




**Obr. 2.** Prostredie nekontrolovaného vývoja softvéru[3]

Dobre plánovaný projekt sa skladá z množstva procesov. Ideálne by bolo, keby boli tieto procesy realizovateľné jeden za druhým ako na obrázku 3, kde je vyobrazený jednoduchý vodopádový model vývoja. Takýto model predpokladá úspešné ukončenie predošlej fázy bez potreby návratu. V skutočnosti sú jednotlivé fázy previazané a navzájom sa ovplyvňujú aj v opačnom smere. Navyše, ak máme veľký tím, nemôže sa každý zúčastňovať na vždy rovnakom procese. Je potrebné uvažovať o paralelizme jednotlivých procesov a subprocesov, aby sme zbytočne nemrhali zdrojmi. Vo väčších firmách sa tento problém dá riešiť prideľovaním na jednotlivé projekty, a tým efektívnejšie využívať ľudské zdroje. Ak však uvažujeme iba jeden projekt, je problém paralelizmu procesov a efektívneho využívania zdrojov pri vytváraní plánov veľmi aktuálny.

Samotný plán musí zakomponovať určité množstvo abstrakcie. Ak by sme chceli naplánovať každý aspekt vývoja, stal by sa plán komplikovaný a neprehľadný a tým nepoužiteľný. Navyše, výraznejší problém v najnižších procesoch by mal za následok lavínovitú zmenu veľkého množstva ďalších procesov. Úprava takéhoto plánu by bola príliš náročná, nehovoriac o tom, že samotné vytvorenie plánu by bolo nadmieru zložité.

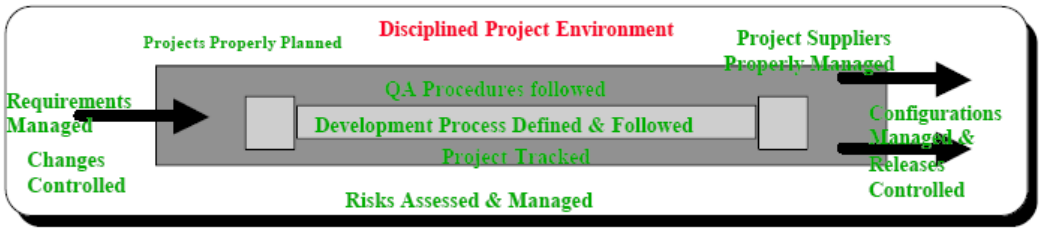



**Obr. 3.** Idealizovaná následnosť hlavných vývojových procesov[2]

Na obrázku 4 je znázornené dobre spravované prostredie vývoja softvéru s dôrazom na korektné plnenie plánov. Požiadavky na produkt sú spracované a spravované. Ak zákazník požaduje funkcionalitu navyše, máme možnosť ohradiť sa špecifikáciou požiadaviek a vývoj novej funkcionality zakomponovať ako zmenu produktu, a tým oddeliť vývoj a financovanie od samotného projektu. Vývoj projektu je sledovaný prideľovaním a kontrolovaním stavu jednotlivých častí vývoja. Výsledkom je systém, o ktorom sa dá povedať, či spĺňa alebo nespĺňa zadefinované požiadavky.

Kontrolované prostredie podľa [3]:

1. len s disciplinovaným manažmentom si udržíme dobré technické praktiky počas krízy
2. disciplína manažmentu a procesov posilňuje technické procesy a personál




**Obr. 4.** Prostredie kontrolovaného vývoja softvéru[3]

## Potreba plánovania

Vytvoriť dobrý plán je skutočne náročné. Človek poverený tvorbou plánu by mal mať skúsenosti s touto problematikou. Navyše musí mať odhad, ktorý sa s praxou zlepšuje. Preto je podľa mňa výhodné, ak ma takýto človek aj skúsenosti s plánovaním ako súčasť plánu, t.z. stretol sa s plánovaním ako programátor, pričom sa na plánovaní nepodieľal.

Myslím si, že každý z nás si v nejakej forme plánuje svoju činnosť. Či už sa jedná o školské záležitosti, ako napríklad zadania, učenie sa na písomky alebo naplánovanie dovolenky a pracovných povinností. Preto je ťažké hovoriť o neplánovaných projektoch. Osobne si myslím, že sa skôr jedná o multiplánový projekt bez synchronizácie plánov. Každý vývojár má svoj plánik, ako vyrieši svoj subproblém vo svojom určenom čase. Ako vníma sebou určené termíny je však pomerne kritické. Ak nemá problém s posúvaním termínov, tak sa ľahko môže stať, že bude riešiť viac problémov naraz a v krátkom časovom spektre. Preto je zadefinovanie požiadaviek a podmienok splnenia dôležitým aspektom plánovania. Osobné plány sa podriadia globálnemu, samozrejme, v reálnych podmienkach nastávajú problémy, a tým sa vývoj sprehľadní a je jednoduchšie určiť sled úloh. Preto si myslím, že plánovanie v tímoch väčších ako 10 ľudí, kde sa zamerania jednotlivých členov prekrývajú, je viac než potrebné ak chceme prehľadný a efektívny priebeh softvérových procesov. Aj pre samotných vývojárov je prínosné, ak vedia aký je plán, akých procesov sa zúčastnia a čo konkrétne treba riešiť. Navyše, tak majú možnosť sa k plánu vyjadriť, a tým ho aj vylepšiť.

Okrem iného plánovanie umožňuje spravodlivejšie ohodnotenie členov tímu konkrétnou špecifikáciou úlohy. To nám dáva možnosť zhodnotiť prácu a výsledky, ktoré daný člen dosiahol. Tým môže plánovanie dokonca prispieť aj v rovine medziľudských vzťahov.

## Potreba adaptácie

V ideálnych podmienkach sa nám stačí držať dobrého plánu. Avšak niekedy sa situácia zmení neočakávane. Gallardo-Valencia a Sim[1] poukazujú na výskyt takejto zmeny, napríklad pri plánovaní cesty autom. Pomocou rôznych plánovacích nástrojov alebo mapy si vyberieme optimálnu cestu z miesta A do miesta B. Ak je však časť tejto cesty opravovaná a nedá sa prejsť, náš plán zlyháva. Podobné problémy sa vyskytujú aj v softvérových procesoch. Napríklad ak dôležitý člen tímu vážne ochorie a pod. Vtedy je potrebné zmeniť plány ako v analogickej ceste autom.

Gallardo-Valencia a Sim[1] vytvorili štúdiu, v ktorej sa snažili zistiť do akej miery sa softvérový vývojári držia plánov. Vývojári tvrdili, že sa nedržia vývojových procesov, hlavne ak používali neformálne procesy. Gallardo-Valencia a Sim[1] zistili, že v skutočnosti sa vývojových procesov držia, aj keď súčasne improvizovali. Tento postup plánovania a improvizácie je pre človeka bežný. Väčšina pohľadov na softvérový proces je však podľa [1] v rozpore s týmto postupom. Softvérový inžinieri si pod pojmom proces predstavia formálny, čiže metodický proces. Samotný dojem

proces vzbudzuje dojem, že procedúra je naplánovaná, zdokumentovaná, štruktúrovaná a striktná. Podobne procesy ktoré tieto vlastnosti nemajú, sa považujú za neformálne, čiže ametodické. Gallardo-Valencia a Sim[1] zistili, že softvérový vývojári, ktorí nepoužívali metodický proces tvrdili, že nepoužívajú žiadny proces. Došli však k záveru: improvizácia je indikátor procesu. Tam kde je improvizácia je aj proces, ktorý sa nedodržuje. Softvérový vývojári teda súčasne používali ametodický proces a vylepšovali ho improvizáciou.

Gallardo-Valencia a Sim[1] následne zadefinovali metodický a ametodický proces:

1. metodický
   - kontrolovateľný – je možné manažovať časti softvéru alebo procesu získané dekompozíciou
   - sekvenčný – pozostáva zo zoradených fáz ktoré musia lineárne nasledovať
   - univerzálny – môže byť aplikovaný na rôzne organizačné aspekty a byť rovnako úspešný
   - cieľovo zameraný – tento typ procesu je založený na predpoklade, že boli definované nejaké ciele, ktoré majú byť dosiahnuté po aplikovaní procesu
2. ametodický
   - oportunistický – javí sa ako následok aktuálnych a nepredvídateľných situácií
   - fragmentovaný – nemôže sledovať lineárnu následosť
   - ad-hoc – závisí od lokálneho rozpoloženia organizácie
   - sociálny – ľudia sú dôležitým aspektom v tomto procese

Gallardo-Valencia a Sim[1] sa zaoberali štúdiou, v ktorej zisťovali, aké typy procesov sa používajú v skutočnosti. Takisto ich zaujímala prítomnosť improvizácie. Subjektom štúdie bolo 8 ľudí zamestnaných v IT firmách, kde sa aktívne zúčastňovali na vývoji. Jeden zo záverov tejto štúdie je aj poznatok, že improvizácia je potrebná nielen v metodických, ale aj ametodických procesoch.

Osobne si myslím, že je vhodnejšie použiť metodické procesy vo vývoji väčších projektov, kde spolupracuje viac ľudí. V menších projektoch a tímoch môže byť prínosom riešiť problémy ametodickými procesmi. Často sa však tieto procesy prelínajú, najmä v neočakávaných situáciách, keď treba niektoré problémy riešiť ad-hoc a následne upraviť aj samotné plány. a tým opäť naviazať na metodický proces. Improvizácia sa ukazuje ako dôležitý prvok vo vylepšovaní metodických softvérových procesov. Takýto postup je človeku vlastný. Držať sa nejakých postupov, ale popritom skúšať aj niečo nové, od čoho si sľubuje vylepšenie. Ak by sme sa držali len už predurčených procesov, napredovanie ľudstva by bolo veľmi ťažkopádne.

## Záver

V tomto dokumente som sa snažil poukázať na problematiku plánovania a adaptácie v softvérových procesoch ako aj na vhodnosť týchto paradigiem. Dalo by sa povedať, že optimálne riešenie sa nedá formálne zadefinovať. Nie vždy je vhodné použiť postupy, ktoré sa inde osvedčili, aj v našom probléme. Optimálne riešenie treba uvažovať nad aktuálnym problémom a aj tak sa môžu vyskytnúť nepredvídané okolnosti. Preto si myslím, že je potrebné voliť rozumnú strednú cestu a nie sa príliš špecifikovať ani na udržiavanie plánu, ale na druhej strane sa ním riadiť. Plán je potrebné brať ako záväzný rozvrh, ktorý však musí počítať s nepredvídateľnými okolnosťami a nechávať si pre ne určitú rezervu. Na tvorbu dobrého plán však potrebujeme dobrých ľudí, ktorí majú schopnosť predvídať a vedia dobre odhadnúť časovú náročnosť jednotlivých problémov a fáz. Schopnosť adaptácie je takisto veľmi dôležitá pre úspešne ukončenie projektu. Ako som už naznačil v úvode, myslím si, že plánovanie a improvizácia nám môžu pomôcť zlepšiť softvérové procesy a samotný vývoj softvéru ako celku. Dôležitý je ale ľudský faktor, ktorý rozhoduje o miere použitia jednotlivých paradigiem.

## Slovník pojmov

- SEI – Software Engineering Institute, vládne financované výskumné a vývojové stredisko, ktoré sa zaoberá výskumom softvérového inžinierstva
- IEEE – Institute of Electrical and Electronics Engineers, vedúca organizácia profesionálov zaoberajúca sa pokrokmi v technológii

## Použitá literatúra

1. Gallardo-Valencia, R.E., Sim S.E.: Planing and improvisation in software process. In: *Crossroads, the ACM Student Magazine,* ACM, 2007
2. Royce, W.W.:Managing the developement of large software systems, WESCON, 1970
3. Zahran, Sami: Software process improvement. Using the Capability Maturity Model, 1998

## Annotation

*Planning and adaptation needs in software process*

Planning of development processes can be nowadays considered as necessity, at least in larger software products. Without a good plan, the development tends to become chaotic what results in reduced efficiency and increased time requirements. Creation of good plan is difficult matter which requires experiences and skills. Even the best plan cannot assure successful development.

Unpredictable occasions may radically affect development process. This document aims to show the need of planning and adaptation of plans, according to unexpected circumstances.