Tvorba plánov

DÁVID KOVÁČ

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

Abstrakt. Plánovanie je jednou z najdôležitejších častí pri tvorbe softvérového projektu. Na začiatku je potrebné analyzovať vytváraný projekt a stanoviť ciele a požiadavky, na základe ktorých sa vytvorí plán činností. Tieto činnosti sa rozdelia medzi jednotlivých členov tímu. Tento plán však nemôžeme považovať za finálny, vzhľadom na to, že počas vývoja vznikajú nové požiadavky, ciele a problémy, ktoré sa na začiatku vývoja neočakávali. To, aby daný projekt správne napredoval, je potrebné dodržiavať stanovený plán a termíny. V tejto eseji sa budem hlavne venovať tvorbe plánov a to porovnaniu modelov prístupu k tvorbe plánov. Tiež okrajovo spomeniem plnenie a aktualizáciu plánov.

Úvod

Plánovanie je každodennou činnosťou každého z nás. Každý deň si plánujeme kedy pôjdeme do školy, do práce a kedy sa z nej budeme vracať. Na aký autobus nasadneme alebo či dnes radšej pôjdeme peši. Každý človek si v podvedomý tvorí akýsi plán, ktorý si v podstate ani neuvedomuje, že to v skutočnosti plán je. Robí to akosi automaticky. Existujú však aj plány, ktoré je potrebné si poriadne dopredu premyslieť a pouvažovať nad nimi, pretože by mohli mať neočakávané následky. Napríklad, študent pred skúškou si musí dobre premyslieť, či mu stačí iba jeden alebo až dva týždne na prípravu. Podobne to funguje aj pri softvérových projektoch, kde je potrebné dôkladne stanoviť plán činností a plán trvania jednotlivých etáp.

Etapy pri vývoji softvérového projektu (Špecifikácia, Analýza, Návrh, Implementácia, Testovanie, Údržba) sú „základným kameňom“ každého softvérového projektu. V týchto etapách sa definujú požiadavky a ciele, ktoré sa následne analyzujú. V ďalšej etape sa navrhne architektúra systému, ktorá sa následne implementuje. Na záver sa vykoná testovanie vytvoreného systému. Každá z týchto etáp musí mať presne stanovené činnosti a termíny. Tieto činnosti sa môžu vzhľadom na použitý model životného cyklu meniť. Životný cyklus teda definuje jednotlivé činnosti, ktoré je potrebné vykonať. Z každej etapy životného cyklu by mali byť hmatateľné výstupy.

Pri nedodržaní, prípadne neplnení plánov sú tieto výstupy oneskorené, respektíve žiadne. Preto je dôležité identifikovať príčiny, prečo sa daný plán nedodržal a snažiť

Manažment v softvérovom inžinierstve, október 2007, s. 1-1.

sa v budúcnosti obmedziť vznik takýchto situácií. Ak však už takáto situácia nastala, neostáva nám nič iné ako opäť analyzovať situáciu a aktualizovať plány.

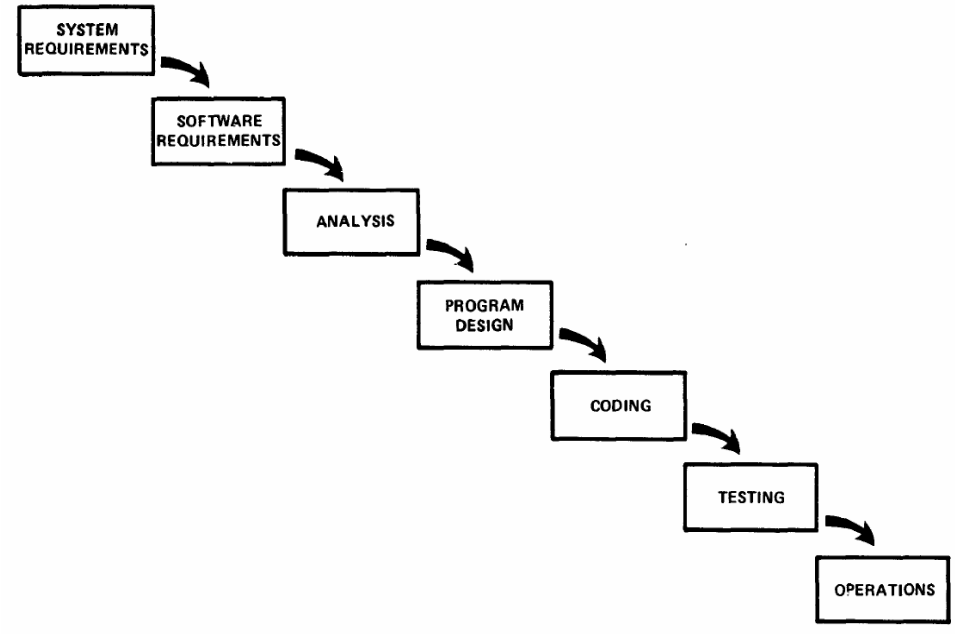
Modely prístupu k plánovaniu

Plán vo všeobecnosti je nejakým rozvrhom činností, ktoré má daná osoba alebo skupina osôb vykonať. Pri softvérovom projekte však treba uvažovať aj o etapách vo vývoji softvéru. Sú to vlastne väčšie časové obdobia, ktoré „zoskupujú“ činnosti s rovnakým zameraním (napr. Analýza, Implementácia). Tieto etapy sú definované pomocou životného cyklu softvérového projektu. Rozvrh činností je potom vlastne dôsledkom životného cyklu tohto projektu. Modelov životného cyklu softvérového projektu je viac. Ja sa však v tejto eseji budem venovať iba niektorým.

Vodopádový model (z angl. „waterfall model“) opísal v roku 1970 doktor Wilson W. Royce. Pôvodne tento model slúžil len ako jeden z prvých krokov k vytvoreniu iteratívneho modelu životného cyklu softvérového projektu. Už sám autor spomína vo svojej analýze, že takýto model je predurčený k zlyhaniu [1]. Zo samotného obrázka (viď **Obr. 1**) môže čitateľ pri menšom zamyslení pochopiť, že takýto model môže byť funkčný iba pre neomylných ľudí. To znamená, že nie je potrebný návrat k predošlým etapám životného cyklu, pretože sme ich dokonale zvládli. Verilo sa tomu, že sme schopný vytvoriť perfektný softvér hneď na prvý pokus. Vzhľadom na to, že každý robíme chyby, nie je možné takýto model akceptovať, pretože nepripúšťa návrat k predošlým etapám vývoja softvéru. Takýto model nie je flexibilný, pretože nedokáže v neskorších etapách reagovať na zmenu požiadaviek a tiež na chyby, ktoré mohli vzniknúť. To sa môže vypomstiť na vzniknutých nákladoch, a preto je vhodné vytvoriť taký model, ktorý minimalizuje náklady, ktoré by mohli vniknúť pri chybách v jednotlivých etapách modelu životného cyklu. I keď sa tento model zdá nepoužiteľný a nefunkčný, stal sa, možno práve pre jeho jednoduchosť obľúbeným modelom životného cyklu softvérového projektu.

Pôvodný Royceov model tvorí sedem nezávislých etáp:

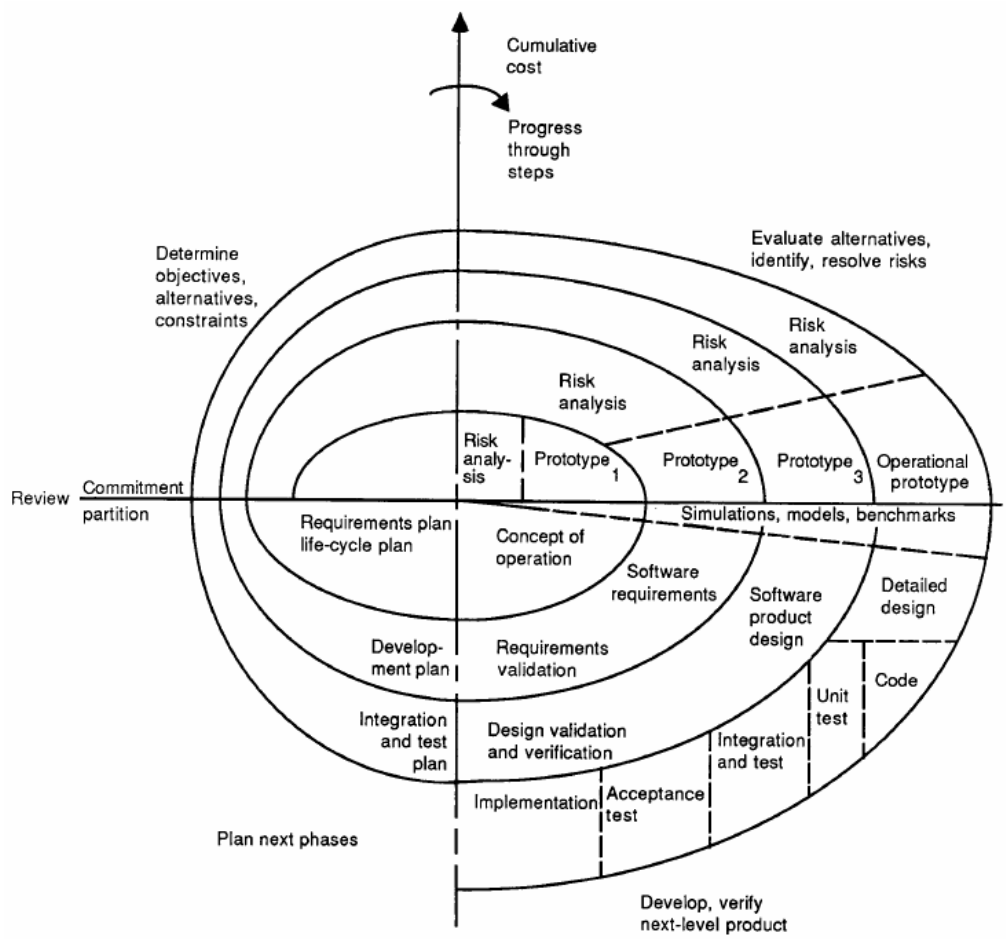
1. Systémové požiadavky – definovanie požiadaviek na systém.
2. Softvérové požiadavky – definovanie požiadaviek na softvér.
3. Analýza.
4. Návrh – podrobný architektonický návrh (koncepcia) systému.
5. Implementácia – vytváranie softvéru.
6. Testovanie – testovanie vytvoreného softvéru.
7. Údržba – operácie spojené s prevádzkou systému.



Obr. 1. Vodopádový model životného cyklu softvérového projektu [1].

Špirálový model [3](z angl. „spiral model“) definoval Barry W. Boehm v roku 1988 v jeho článku „A Spiral Model of Software Development and Enhancement“. Tento model vznikol na základe vodopádového modelu a jeho neskorších úprav. Vzhľadom na tento fakt je dobré sa na tento model pozerať ako na vodopádový model, v ktorom každý cyklus je predchádzaný analýzou rizík a jeho vykonaním. Tento model už vylepšuje nedostatok vodopádového modelu a to tým, že zavádza analýzu rizík. Rizikami sú nepriaznivé okolnosti, ktoré môžu nastať počas vývoja softvéru. Môže to byť strata zamestnanca, voľba nevhodného (nepoužiteľného) softvérového prostriedku a podobne. Špirálový model sa skladá zo štyroch stavov (z angl. „stages“), ktoré môžete vidieť aj na obrázku (viď **Obr. 1**):

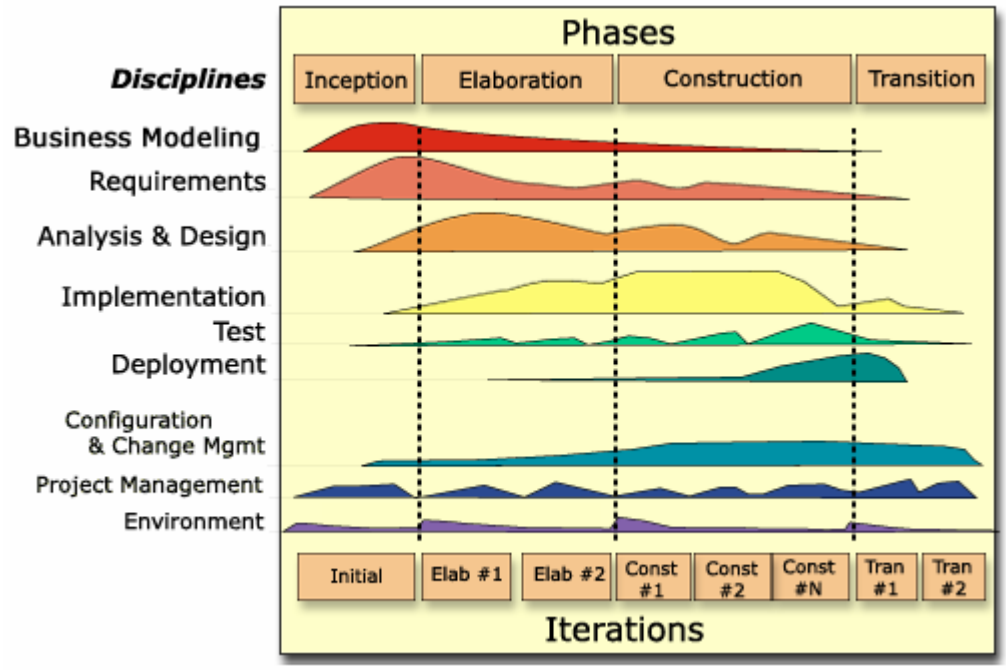
1. Určenie cieľov, alternatív a obmedzení.
2. Vyhodnotenie alternatív, identifikovanie a vyriešenie rizík – mali by sa analyzovať riziká priradené rozhodnutiam z predchádzajúceho stavu. Počas tohto stavu sa môže urobiť prototyp alebo vykonať simulácie softvéru.
3. Vývoj, overenie produktu ďalšej úrovne – pozostáva z vývoja, špecifikácie návrhu, kódu a overenia (testovania).
4. Vyhodnotenie a plánovanie ďalšej fázy – znamená revíziu predchádzajúcich stavov a plánovanie ďalšej fázy vývoja. Na základe výsledkov z predchádzajúcich stavov sa môžeme rozhodnúť pokračovať s vodopádovým modelom alebo evolučným modelom.



Obr. 2. Špirálový model životného cyklu softvérového projektu [3].

Na obrázku (**Obr. 2**) znázorňuje polomer špirály vzrastajúce náklady na vývoj softvéru a uhol, ktorý v danom bode špirála vytvára zobrazuje pokrok vývoja. Na začiatku vývoja sa tento model životného cyklu zameriava hlavne na analýzu rizík a vytváranie prototypov, čo vedie k značnej redukcii rizík, ktoré by sa mohli objaviť v neskorších fázach vývoja. Výhodou tohto prístupu je, že náklady, ktoré sa investujú do vývoja sú priamo úmerné vytváranému softvéru. Napríklad, ak sa vo vývoji softvéru s vodopádovým modelom životného cyklu objavili v etape implementácie nové neočakávané skutočnosti, je potrebné (vo väčšine prípadov) urobiť novú analýzu a takisto návrh, čo však už nemusí byť lacnou záležitosťou. Naopak, u špirálového modelu sa tomuto javu snaží predchádzať práve inkrementálnou tvorbou prototypov a analýzou rizík.

Rational Unified Process (RUP) bol vytvorený Rational Software Corporation. Popisuje všestranný životný cyklus so štyrmi fázami (Inception, Elaboration, Construction, Transition), z ktorých každá má špecifické ciele a míľniky (viď **Obr. 3**). Ak sa vo fáze Inception nedosiahne míľnik nazývaný „Lifecycle Objective Milestone“, je môžné projekt ukončiť alebo sa urobí nový návrh, ktorý spĺňa dané kritériá tejto fázy. Fáza Elaboration má míľnik nazývaný „Lifecycle Architecture Milestone“, pri ktorom ak nedosiahneme požadované kritériá, stále je možné projekt ukončiť alebo urobiť nový návrh. Vo fáze Construction sa sústredí na vývoj jednotlivých súčastí vytváraného systému. V tejto fáze sa produkujú prvé externé vydania softvérového projektu. Ukončenie tejto fázy sa nazýva „Initial Operational Capability Milestone“. Poslednou fázou je fáza Transition, v ktorej sa vyprodukovaný systém „doručí“ užívateľovi. Táto fáza zahŕňa zaúčanie užívateľov, údržbu a beta testovanie. Ak sú všetky ciele splnené, tak sa dosiahol posledný míľnik zvaný „Product Release Milestone“.



Obr. 3. RUP architektúra procesov [4].

V rámci každej z týchto fáz sa nachádzajú malé iterácie (viď **Obr. 4**). RUP je vlastne implementáciou špirálového modelu. Tieto iterácie umožňujú vylepšovanie danej fázy projektu. Napríklad, ak sa nachádzame vo fáze Inception, tak do fázy Elaboration môžeme vstúpiť len ak sme splnili kritériá potrebné na dosiahnutie daného míľnika, čomu však mohlo predchádzať aj niekoľko iterácií v danej fáze.

Obr. 4. Životný cyklus RUP [5].

Vzhľadom na to, že špirálový model a RUP sú priamym alebo nepriamym vylepšením vodopádového modelu, je jednoduché dedukovať, že oba modely sú prijateľnejšie oproti pôvodnému vodopádovému modelu. Avšak, podľa môjho názoru, záleží od projektu, v ktorom sú tieto modely použité. RUP a špirálový model sú jednoznačne určené pre veľké softvérové projekty, zato vodopádový model môže byť použitý v malých softvérových projektoch, kde nie je potrebné vynaložiť toľko úsilia do plánovania. Ďalším možným využitím vodopádového modelu je možnosť používať ho ako zjednodušený model životného cyklu softvérového projektu pri výučbe softvérového inžinierstva.

Plnenie a aktualizácia plánov

Plnenie plánov je asi jedným z najväčších problémov pri vývoji softvéru. Na prvý pohľad sa môže zdať, že problém je iba v členoch softvérového tímu, ktorí nie sú schopný plniť si svoje povinnosti včas a v danom termíne. Existuje však aj mnoho iných dôvodov, prečo softvérový projekt napreduje pomaly, respektíve stagnuje. Prvým aspektom, ktorý môže ovplyvniť plnenie plánov je samotná tvorba plánov. Plán ako taký môže byť pesimistický, tj. manažér plánovanie má pesimistický pohľad na celý svoj vývojový tím a celkovo i na samotný projekt. Termíny, ktoré sú stanovené pre dané etapy životného cyklu sú ďaleko presahujúce časy skutočne potrebné pre danú etapu vývoja. Ďalším prístupom k tvorbe plánov je optimistický pohľad, kedy manažér plánovania stanovuje termíny, ktoré sú len veľmi ťažko dosiahnuteľné v danom čase. Síce je možné, že sa zvýši produktivita, ale vo väčšine prípadov to bude viesť nedodržiavaniu termínov a neustálej aktualizácii plánov. Najideálnejším pohľadom na tvorbu plánov je realistický pohľad, kedy manažér vie reálne odhadnúť čas potrebný na realizáciu jednotlivých činností v daných etapách životného cyklu softvérového projektu. Ďalším aspektom môžu byť neočakávané udalosti ako napríklad dlhodobá neprítomnosť členov softvérového tímu, nesprávna voľba podporných prostriedkov alebo už spomínaná nezodpovednosť členov tímu.

Dôsledkom odchýlok od ideálnej tvorby plánov alebo neočakávaných udalostí je aktualizácia plánov. Plány sa musia nanovo vytvoriť pre jednotlivé etapy, ktorých sa táto zmena týka. Je však možné týmto zmenám aj do určitej miery predchádzať, napríklad ak zamestnancov motivujeme finančnými odmenami alebo inými výhodami, ktoré by viedli k dodržaniu termínov. Samozrejme najdôležitejšie správne tvoriť plány.

Záver

V mojej práci Tvorba plánov som sa venoval modelom prístupu k tvorbe plánov, kde som opísal tri modely životného cyklu softvérového projektu. Ako prvému som sa venoval vodopádovému životnému cyklu. Tento model som označil ako aj sám autor ako nefunkčný, pretože očakáva neomylnosť pri tvorbe softvéru. Ďalší model, ktorému som sa venoval bol špirálový model. Tento model vznikol rôznymi úpravami a vylepšeniami vodopádového modelu. Sústredí sa najmä na podrobnú analýzu rizík a tvorbu prototypov. Tento model sa stal aj základom pre Rational Unified Process (RUP). Podľa môjho názoru sa všetky tri modely môžu uplatniť v reálnom živote, záleží len od typu softvérového projektu. RUP a špirálový model sú určené pre väčšie softvérové projekty. Naopak vodopádový model nájde uplatnenie pri menších projektoch, respektíve v školských učebniciach ako zjednodušený model životného cyklu softvérového projektu.

V ďalšej časti som sa venoval plneniu a aktualizácii plánov. Uviedol som niekoľko dôvodov prečo plány zlyhávajú a ako je možné čiastočne predchádzať aktualizácii plánov.

Použitá literatúra

1. Royce, W.W.: Managing the Development of Large Sofware Systems. *Proc. of IEEE Western Conference (Wescon)*, 1970. (dátum prístupu 22.10.2007)
2. http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf
3. Boehm, W.B.: A Spiral Model of Software Development and Enhancement, *IEEE Computer*, 61-72, Máj 1988.
4. Wikipedia: IBM Rational Unified Process, (dátum prístupu 24.10.2007) http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process.
5. http://era.nih.gov/docs/rup_fundamentals.htm, (dátum prístupu 24.10.2007)

Annotation

Project Planning

Planning is one of the most important parts in development of software project. At the beginning it is important to analyze generated project and to determine goals and requirements, which are needed for creating the plan of activities. These activities are divided between team members. However we can not consider this plan as final, because during development rise new

requirements, goals, problems, which were not expected at the beginning. It is necessary to follow defined plan and terms in order to successful continuity of the project. In this work I will analyze plan creation, comparing the models of software lifecycle. I will also mention admission and actualization of plans.