

Klasický management

udržování a rozvíjení zavedených systémů, které jsou prostředkem pro nepřetržitou, kontinuální a opakující se tvorbu požadovaných výstupů.

Projektový management

slouží k zabezpečení realizace jedinečných, neopakovatelných, časově a zdrojově limitovaných procesů, které vedou k dosažení předem stanovených cílů.

Management projektu

- Plánování projektu
- Organizování a koordinování projektů
- Koordinování projektů
- Řízení realizace projektu
- Vytváření organizačního prostředí

Projekt

znamená plánování a řízení rozsáhlých operací vedoucích ke konkrétnímu cíli, se stanovenými termíny zahájení a ukončení, s omezenými zdroji a náklady.

Není to periodicky se opakující rutinní činnost, ale jedinečná, systémová činnost s nejistotou a rizikem.

Plánování

je popis (nikoli toho, co se stane, ale) toho, co chceme, aby se stalo.

Řízení realizace

je proces, kterým chceme dosáhnout toho, aby se plánované události skutečně staly, aby nedocházelo k neplánovaným událostem

Proces plánování projektu

- stanovení cílů a definování strategie vedoucí k jejímu dosažení
- zpracování strukturované dekompozice činností projektu
- vytvoření projektové organizační struktury a sestavení projektových týmů
- zpracování implementačních plánů projektu, tj. časových plánů, plánů nákladů, alokace zdrojů
- specifikace nástrojů a technik pro řízení projektu
- identifikace možných omezení, rizikových oblastí a návrh způsobů eliminace těchto vlivů

Proces řízení realizace projektu

- realizace implementačních plánů projektu a koordinace subjektů podílejících se na jeho realizaci
- identifikace a analýza aktuálních dat
- řízení, kontrola a průběžné vyhodnocování, analýza a korekce průběhu projektu – kontrola stanovených cílů, termínů a čerpání zdrojů a nákladů
- řešení konfliktních a nestandardních situací
- technická a administrativní podpora projektu
- změnová řízení
- koordinace postupné integrace systému
- vyhodnocení dílčích etap projektu a návrh úprav

Začlenění projektového managementu do organizační struktury

- útvarový projektový management
- čistý projektový management
- maticový projektový management
- síťový projektový management

Týmový management projektu

projektový tým - projektová hierarchie:

- manažer projektu
- členové týmu
- expertní tým
- vedoucí projektové skupiny
- dozor projektu

Organizace týmu (styly řízení) - metody týmové práce:

- pracovní porady
- psychologie týmové práce:
- motivace
- rizika skupinového myšlení
- komunikace a koordinace
- řešení konfliktů

Manažer projektu zodpovídá za:

- řízení realizace implementačních plánů
- identifikace odchylek od plánů, včetně návrhů a realizace nápravných opatření
- poskytování informací o průběhu realizace projektu
- formulování a předkládání požadavků, která jsou nad rámec jeho pravomoci
- předvídání vzniku problémů a hledání vhodných způsobů jejich řešení
- vyřizování pracovních nároků a problémů členů týmu
- sledování a vyhodnocování vynaložených nákladů vzhledem k danému rozpočtu
- vytváření potřebných pracovních kontaktů na všech úrovních řízení

Možné charakteristiky členů týmu

- neví a neví, že neví (neví a tvrdí, že ví)
- neví a ví, že neví
- ví a ví, že ví
- neví, že ví

Asertivní chování = sebeprosazování otevřenou komunikací při zachování práv druhých.

Synergický efekt = zesilující účinek projevující se např. v týmové práci.

Brainstorming = skupinová metoda hledání alternativ.

Pracovní porady

(pravidelné porady, porady zaměřené na kvalitu, koordinační porady, plánovací porady, informační porady..)

- kontrola postupu prací a specifikace důsledků neočekávaných změn
- diskuse o alternativních možnostech realizace projektových činností
- udržování potřebné informovanosti členů týmu
- koordinování potřeb projektu s dalšími stranami
- kontrola a minimalizace projektových nákladů
- dodržování kvality
- řešení konfliktních situací

Zásady pro efektivitu porad

- včas poskytnout všem pracovní materiály, které se budou projednávat
- znát předem čas zahájení a ukončení porady
- zahájit porady bezodkladně
- porada musí mít svého předsedajícího a zapisovatele
- zápis má obsahovat uložené úkoly jmenovitě a s termínem
- na závěr porady je třeba zrekapitulovat úkoly a termíny
- zápisy se musí distribuovat podle jasných pravidel (komu a do kdy)
- zvát na porady jenom ty pracovníky, kterých se daná problematika týká

Motivace

motivační profil člověka hodnotové skupiny:

- individuální (spokojený život, rodina, domov)
- pracovní (sebezdokonalování, seberealizace)
- společenské (kontakt s lidmi, postavení)
- materiální (peníze, jídlo, zábava)
- hodnota volného času (život podle vlastních zálib)
- pracovníci orientovaní na úkol (motivovaní samotnou prací)
- pracovníci orientovaní na spolupráci (motivovaní přítomností a prací kolegů)
- pracovníci orientovaní na sebe (motivovaní vlastním úspěchem)

Zásady neegoistického programování:

- chyby v programech se považují za nutné zlo
- programy jsou považovány za společné dílo týmu, nikdo nepovažuje program za vlastní dítě, které je třeba hájit
- při rozhodování je každý ochoten přijmout řešení optimální pro celý tým, i když to může znamenat dočasnou nevýhodu pro něho samého

Výhody menších týmů (2-8 členů):

- snazší dohoda norem kvality programů, jak mají být psány, testovány a předávány
- možnost se učit jeden od druhého
- snáze se realizuje neegoistické programování
- znají navzájem svou práci, není takový problém, když někdo odejde

Vhodné pracovní podmínky pro programátory:

- dostatek soukromí - možnost pracovat v klidu bez vyrušování
- možnost pracovat při denním světle
- programátoři jsou vyhraněné osobnosti a programování je individuální práce, vyplatí se dát jim možnost upravit si pracoviště a pracovat tak, jak jim vyhovuje (při splnění zákonných podmínek).
- důležité je, aby se tým měl kde scházet.

Projektový management (PM) vs. Management projektu (MP)

Teorie

PM =souhrn organizačních, logistických a strategických opatření podniku, podle kterých se každý projekt IT provádí. Všeobecně "slouží k zabezpečení realizace jedinečných, neopakovatelných, časově a zdrojově limitovaných procesů, které vedou k dosažení předem stanovených cílů"

PM =standardně vypracován, dokumentován, je přístupný každému pracovníku v IT a je pravidelně aktualizován

PM =předmětem zaškolení všech pracovníků v IT

Od **PM** se očekává, že zoptimalizuje přípravu, plánování, provádění projektu a jejich dovedení k úspěšnému konci

PM =vychází ze zkušenosti, že při provádění projektu se narazí na rizika, že se vyskytnou problémy a že dochází k projektovým změnám

MP - se vztahuje na konkrétní projekty, které pomaha planovat, provadet a zakončovat

MP - odvozuje pracovní postupy ze standardního PM

MP - je uplatňován na veškeré, v PM definované činnosti, které mají charakteristiku odpovídající definici projektu

MP - konkretizuje a upřesňuje procesy popsané v PM

MP - má zpětnou vazbu na PM – poskytuje konkrétní informace pro aktualizaci PM

Praxe

(-) **PM a MP** se mnohdy nerozlišují ("... vždyť je to totéž")

(-) Neexistuje žádná standardní dokumentace ani pravidelná aktualizace PM

(-) PM jako standardní dokumentace sice existuje, ale je většinou pracovníků IT neznámá, neboť žádné zaškolení se nekonalo

(-) "Jak to, že jste ten projekt nezvládli v předepsaném čase, vždyť na to máme PM?" Pouhé očekávání bez přičinění ale nevede k požadovanému cíli

(-) Rizika se neberou vážně či vůbec v úvahu, problémy se řeší až když se vyskytnou a "žádné velké změny nebudou, pokud se vše perfektně naplánuje" - toto mínění je pořád dost časté, takže se těmto oblastem v dokumentaci PM nevěnuje patřičná pozornost

(-) Snaha uplatnit MP na jinou než projektovou činnost (např. rutinní práce) nemá smyslu, protože charakteristika těchto prací uplatnění MP neumožňuje

(-) **MP** se chápe jako pracovní náplň vedoucího oddělení, který coby manažer projekty ve svém oddělení sám spravuje a řídí

(-) "V **PM** je sice definováno, že nejkratší doba trvání projektu je 1 měsíc, což odpovídá době trvání naší práce, ale žádný MP dělat nebudeme, to se nevyplatí". Tento názor je nesprávný a vede mj. časem k znehodnocení celého PM

(-) **MP** má u různých projektů různou, někdy i protichůdnou náplň v závislosti na vedoucích projektu, kteří zásady PM neuplatňují

(-) Po dokončení projektu se projekt nevyhodnocuje (častý argument: "na to teď není čas"), takže **MP** nedodá k dispozici informace o průběhu projektu, které jsou důležitýmj. k aktualizaci všeobecné metodiky popsané v PM

...

Synoptické srovnání ‚vedoucí projektu‘ vs. ‚vedoucí oddělení‘

vedoucí projektu

moderator, coach
generalista
menici se ulohy
zpochybňuje
prolinave mysleni
globalni reseni
inovacni postupy

vedoucí oddělení

manazer
specialista
rutinni ulohy
zastupuje existujici
linearni mysleni
specificka reseni
klasicke metody

myslenky zavazují
orientace na cíl, řešení

pravidla dominují
nejlepší plnění funkce

Co potřebuje projekt

- HW a SW
- Case nástroje
- Počítače na testování, školení ...
- Možnosti kopírování
- Testery
- Návrháři
-

- Počítačovou síť a metody komunikace
- SW s kterým má navrhovaný spolupracovat
- Nástroje pro dokumentaci
- Programátory
- Managery
- Odborníky na specifikace

Výchozí podmínky

- složitost projektu
- metriky minulých projektů

- velikost projektu
- variabilita v softwarových požadavcích

Popis rozsahu softwaru

- funkce
- rozhraní
- spolehlivost

- chování
- omezující podmínky

Omezení

Programátor je schopen zvládnout maximálně 10000 řádků kódu
Typický WP má ????? řádků kódu

Stanovení rozsahu

Interview se zákazníkem = předběžná řízená schůzka se zákazníkem

1. Context free questions (bezkontextové otázky)

Kdo tu práci požaduje, kdo ji bude užívat, jaký bude ekonomický užitek při úspěšném ukončení, je ještě jiná možnost, jak to vyřešit?

2. Hlubší pochopení problému a názoru zákazníka

Jak byste charakterizoval "dobrý" výstup? Na jaké problémy je toto řešení zaměřeno? Ukažte mi (popište) prostředí, kde to bude systém pracovat? Jsou nějaké speciální požadavky na chování systému a na jeho

3. „Meta otázky“

Jste ta správná osoba, která mi může na tyto otázky odpovědět? Jsou vaše odpovědi oficiální? Jsou mé otázky relevantní k danému problému? Nedávám vám moc otázek? Je tu ještě někdo další, kdo by mohl poskytnout doplňující informace? Je ještě něco, na co bych se měl zeptat?

Další setkání jsou formálnější (řešení problému, vyjednávání a specifikace)

FAST (Facilitated Application Specification Techniques)

- schůzka realizátorů a zákazníků
 - pravidla pro přípravu a průběh
 - používá se tabule, flip chart apod.
 - cílem je identifikovat problém, navrhnout řešení, vyjednat odlišné přístupy a specifikovat předběžnou množinu požadavků
- řízena neutrální stranou (facilitátorem)
 - agenda

Plánování zdrojů

Popis zdroje, jeho dostupnost, kdy bude požadován, na jak dlouho

- lidské zdroje
- HW a SW nástroje
- využitelné SW komponenty

SW komponenty

hotové komponenty (off-the-shelf components)

komponenty se zkušeností (full-experience components)

komponenty s částečnou zkušeností (partial-experience components)

nové komponenty

Při plánování je třeba odhadnout:

- dobu trvání jednotlivých činností
- potřebné zdroje (lidí, HW, SW a jiné zařízení a nástroje)
- pracnost činností (v mm)

Dvě kategorie odhadů:

- dekompozice (rozdělení hlavních funkcí a odhad velikosti nebo pracnosti implementace každé funkce)
 - empirické modely (odvozené formule pro pracnost a čas)
- Přesnější odhady - porovnáním více technik
Vše závisí na dobrých historických datech!

Co je to projekt?

Projekt NENÍ

- projekt je ona složka dokumentace, podle které se něco bude dělat,
 - projekt je vlastně každý nový výrobek, který zavádíme
 - projekt je každý velký úkol, který je v podniku řešen
- _????????

Definice projektu obsažená v ČSN ISO10 006 :

"Projekt je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji".

Projekt

- je jednorázová transformace vstupů (informace, prostředí, materiál, peníze, schopnosti a dovednosti zúčastněných lidí) na výstupy – cílové produkty, za pomoci vývojových činností (uspořádaných do etap, kroků a úkonů) a koordinovaných řídicími činnostmi.
- vždy zaměstnává skupinu lidí a ovlivňuje jiné skupiny lidí.
- projekt je vždy spojen s rizikem neúspěchu - je jedinečný
 - nikdy zcela přesně nevíme, co nás v průběhu jeho realizace čeká nebo zaskočí.

Co je to úspěšný projekt?

Abychom projekt úspěšně realizovali, **musíme jej řídit a uřídit.**

Abychom mohli projekt řídit, **musíme mít** nějaký scénář či osnovu - **plány projektu.**

Aby plány projektu byly použitelné a měly naději na úspěch musí být vytvořena tzv. **strategie projektu.**

A musíme systematicky pracovat na tom, aby se naše schopnosti realizovat projekty zlepšovaly

Musíme se učit „lepší cestě“ - čtyři témata:

- strategie projektu
- plány
- řízení
- zlepšování

Co znamená, že projekt dopadl úspěšně?

- bylo dosaženo definovaných cílů
- nebyl překročen rozpočet
- cíle byly dosaženy v čase, který jsme na začátku předpokládali

Troj-imperativ projektu = úspěch projektu znamená splnění cíle ve třech dimensích

věcně = (**CO** se má udělat),

časově = (**KDY** se to má udělat)

nákladově = (**ZA KOLIK** se to má udělat).

Strategie projektu

- je dohodnut cíl
- vybrána v dané chvíli nejvýhodnější cesta
- analyzován počáteční stav

Čím začít?

- analýzou stávajícího stavu?
- vytipováním možných cest?
- dohodnou o cílech?

- jaký "problém" (v americké kultuře by spíše řekli „challenge - výzvu“) má projekt řešit.

Dokud nemáte jasno, není možné stanovit dobře přínosy, cíle a hlavní výstupy. Teprve po jejich stanovení udělejte analýzu stávajícího stavu.

Logická rámcová matice = Logical Framework Matrix

Tato dnes již poměrně rozšířená metoda formou jednoduché matice umožňuje stanovit přínosy, cíle, klíčové výstupy, metriky a jejich zdroje a vnější předpoklady.

Špatně

Pokud Logická rámcová matice vzniká jako povinný dokument tak, že ji přes noc před odevzdáním

projektového záměru napíše projektový manažer nebo předkladatel projektu, je to naprosto formální a zbytečná práce a lze s plnou vážností říci, že takovýto projekt sdílenou strategii nemá.

Jak přistupovat k analýze rizik

- text, kde jsou formou odrážek volně promíchána rizika, hrozby, scénáře a možné škody
- zdánlivě propracovanou tabulku o mnoha sloupcích a řádcích doporučení ČSN ISO 10 006.

Nebezpečí = potenciální výskyt nepříznivé události

Hrozba = konkrétní projev nebezpečí

Scénář = nepříznivý děj, který hrozba způsobí

Pravděpodobnost = pravděpodobnost výskytu dvojice hrozba-scénář

Škoda = újma způsobená v důsledku nepříznivé události

Plánování projektu - řízení podle plánů

A plánování není nic jiného, než postupná odpověď na správně položené otázky.

Tyto otázky i jejich pořadí (které je rovněž důležité) jsou velmi impresivní:

- CO
- JAK
- S KÝM
- KDY
- ZA KOLIK

Řídit projekt znamená způsobit, že co je naplánováno, bude taky uděláno. Rozpis práce

Soustava projektů

To, čím se projekty ovlivňují je vlastně omezením soustavy projektů.

Úvahy o závislosti činností posuneme o úroveň výš a řešíme závislosti mezi projekty.

U soustavy projektů je třeba:

1. identifikovat omezení soustavy
2. rozhodnout jak omezení využít
3. vše na soustavě podřídit tomuto rozhodnutí
4. pozvednout omezení soustavy

Kritické faktory úspěchu při realizaci projektů

- Mějte jasnou vizi a cíle.
- Vytvořte strategii projektu.
- Sponzor.
- Týmová práce.
- Řešení jako skutečný projekt.
- Definujte funkční řídicí výbor a řešitelský tým.
- Schopnost vidět projekt v souvislostech.
- Princip osobní odpovědnosti.

Řízení rizik

Rizika jsou charakteristická tím, že vždy zahrnují **nejistoty** a v případě, že nastanou, **ztráty**.

Dynamika projektu 'Ideální' stav

- existuje, dokud se projekt myšlenkově připravuje
- vlivy se berou v úvahu, ale ještě nepůsobí

Vnitřní vlivy

- intenzita a dopad na projekt jsou závislé na podnikové kultuře
- nejen vlivy samy, ale i přístup, jak s nimi nakládat, ovlivňuje chod projektu

Vnější vlivy

- je nutné je pokud možno všechny zjistit, aby se jejich dopad dal z kalkulat
- zjištění možnosti jejich oslabení předem přínosem pro chod projektu

PROCES ŘÍZENÍ RIZIK

Rozeznání rizik - Seznam možných rizik

Analýza rizik - Seznam rizik podle priorit

Plánování rizik - Předcházení rizik

Monitorování rizik - Vyhodnocení rizik

Kategorie rizik

Projektová rizika = *Project risks*

Technická rizika = *Product risks*

Obchodní rizika = *Business risks*

Obchodní rizika - příklady

- vybudování skvělého produktu, který nikdo nechce (**marketingové riziko**)
- vybudování produktu, který už nezapadá do obchodní strategie firmy (**strategické riziko**)
- vybudování produktu, kterému obchodní zástupci nerozumí a neví, jak ho prodat
- ztráta podpory vedení, vlivem změny zaměření nebo změny osob (**riziko managementu**)
- ztráta rozpočtu (**rozpočtové riziko**)

Jiné rozdělení

známá rizika - dají se odhalit po pečlivém vyhodnocení plánu, obchodního a technického prostředí a jiné spolehlivé zdroje

předvídatelná rizika - jsou extrapolována ze zkušenosti z předchozích projektů

nepředvídatelná rizika - dají se jen velmi těžko předpovědět

Identifikace rizik

rizika obecná (generic risks)

rizika specifická (product specific risks)

Vytvoření seznamu známých a předvídatelných rizik (*risk item checklist*).

Rozdělení obecných rizik

- | | |
|-----------------------------------|--------------------------|
| 1. velikost produktu | 2. obchodní dopad |
| 3. charakteristiky zákazníka | 4. definice procesu |
| 5. vývojové prostředí | 6. vytvářená technologie |
| 7. velikost týmu a jeho zkušenost | |

1. Rizika velikosti produktu

Odhad velikosti v LOC nebo FP?

Stupeň důvěry v odhad?

Odhad velikosti produktu v počtu programů, souborů, transakcí?

Průměrná procentuální odchylka velikosti produktu podle předchozích produktů?

Velikost databáze vytvářené nebo používané v produktu?

Počet uživatelů produktu?

Počet projektovaných změn požadavků pro produkt -před dodáním, po dodání?

Množství znovupoužitého softwaru?

2. Rizika obchodního dopadu

Jak produkt ovlivní zisk společnosti?

Je produkt viditelný vedením společnosti?

Je dodací termín rozumný?

Počet zákazníků, kteří budou používat tento produkt a konsistence jejich potřeb vzhledem k produktu?

Počet ostatních produktů/systémů, s nimiž musí produkt spolupracovat?

Kvalifikace koncových uživatelů?

Množství a kvalita dokumentace produktu, která musí být produkována a dodána zákazníkovi?

Státní omezení (normy) na konstrukci produktu?

Cena za pozdní dodání?

Cena za defektní produkt?

3. Rizika týkající se zákazníka

Pracoval jsi se zákazníkem již dříve?

Ví zákazník dobře, co chce?

Souhlasí s tím, že bude muset věnovat čas realizátorům pro stanovení požadavků a identifikaci rozsahu projektu?

Chce mít dobré komunikační spojení s realizátorem?

Chce se zúčastnit sledování (inspekci) projektu?

Je zákazník technicky vzdělaný v oblasti produktu?

Nechá zákazník tvoje lidi dělat svou práci -tzn. nebude se jim dívat přes rameno?

Rozumí zákazník softwarovému procesu?

4. Procesní rizika

Otázky procesu

Podporuje vedení společnosti standardizaci procesu softwarového vývoje?

Má vaše organizace metodiku procesu vývoje softwaru, použitelnou pro tento projekt?

Přijali členové týmu metodiku softwarového procesu a budou ji používat?
Byla ona metodika softwarového procesu používána i pro jiné projekty?
Poskytuje vaše organizace kurzy pro manažery a technické pracovníky?
Jsou publikované standardy SI poskytovány všem vývojářům a manažerům?
Jsou formální revize (přezkoumání) specifikací požadavků, návrhu a kódu řádně prováděny?
Jsou formální revize (přezkoumání) testovacích procedur a testovacích případů řádně prováděny?
Jsou formální revize (přezkoumání) dokumentovány, včetně nalezených chyb a použitých zdrojů?
Je zajištěno, aby práce na projektu odpovídaly standardům SI?
Jsou řízeny konfigurace k udržení konsistence v systémových/softwarových požadavcích, návrhu, kódu a testovacích případech?
Jsou řízeny změny v požadavcích zákazníka, které mají dopad na software?
Je pro každý subkontrakt dokumentována práce, specifikace softwarových požadavků a plán softwarového vývoje?
Existuje postup řízení a sledování práce subdodavatelů?

Otázky techniky

Jsou používány facilitativní techniky při komunikaci zákazníka s vývojářem?
Jsou používány specifické metody pro softwarovou analýzu?
Používáte specifické metody pro návrh dat a architektury?
Je více než 90% vašeho kódu napsáno v jazyce vyšší úrovně?
Je definována a používána specifická konvence pro dokumentaci kódu?
Používáte specifické metody pro návrh testovacích dat?
Používají se softwarové nástroje pro podporu plánování a řízení?
Používají se softwarové nástroje pro řízení konfigurací a změn?
Používají se softwarové nástroje pro podporu analýzy softwaru a proces návrhu?
Používají se nástroje pro tvorbu softwarových prototypů?
Používají se softwarové nástroje pro podporu testování?
Používají se softwarové nástroje pro podporu tvorbu dokumentace?
Sbírají se kvalitativní metriky v každém softwarovém projektu?
Sbírají se metriky produktivity v každém softwarovém projektu

5. Rizika vývojového prostředí

Máte nástroj pro řízení management softwarového projektu?
Máte nástroj pro řízení softwarového procesu?
Jsou k dispozici nástroje pro analýzu a návrh?
Poskytují nástroje pro analýzu a návrh metody vhodné pro vytvářený produkt?
Jsou k dispozici překladače nebo generátory kódu a jsou vhodné pro vytvářený produkt?
Jsou k dispozici nástroje pro testování a jsou vhodné pro vytvářený produkt?
Jsou k dispozici nástroje pro řízení konfigurací?
Jsou všechny softwarové nástroje integrovány?
Byli členové týmu vyškoleni v používání všech nástrojů?
Jsou k dispozici experti, schopní konzultovat používání nástrojů?
Máte on-line help a dokumentaci pro ony nástroje?

6. Technologická rizika

Je budovaná technologie pro vaši organizaci nová?
Požaduje zákazník tvorbu nových algoritmů nebo vstupně výstupní technologie?
Bude mít software rozhraní s novým a neověřeným hardwarem?
Bude mít vytvářený software rozhraní s nakoupeným softwarovým produktem, který není prověřený?
Bude mít vytvářený software rozhraní s databázovým systémem, jehož funkce a chování nebylo ověřeno v dané aplikační oblasti?
Je požadováno speciální uživatelské rozhraní?
Je požadována tvorba programových komponent, které jsou odlišné od dříve vytvořených ve vaší organizaci?
Je požadováno používání nových metod analýzy, návrhu nebo testování?
Je požadováno použití nekonvenčních metod vývoje softwaru, jako jsou formální metody, přístupy založené na UI nebo neuronových sítích?
Obsahují požadavky speciálně rozšířená omezení produktu?
Není si zákazník jistý, zda bude proveditelná požadovaná funkčnost?

7. Rizika spojená s velikostí týmu a jeho zkušeností

Máte nejlepší odborníky pro projekt?

Mají tito lidé odpovídající zkušenosti?

Je jich dostatečný počet?

Jsou všichni vyčleněni na celou délku projektu?

Budou pracovat na projektu na plný úvazek?

Rozumí všichni projektu?

Dostali nezbytné školení?

Bude případný úbytek pracovníků dostatečně nízký, aby umožnil kontinuitu?

Není-li s projektem spojeno žádné riziko, nesnažte se projekt řešit, neboť projekty bez skutečných rizik „nemají šanci“, nepřinášejí nikdy zisk a i proto nebyly realizovány už v minulosti.

Rizikové komponenty

Rizika provedení-stupeň nejistoty, že produkt bude odpovídat požadavkům a bude vyhovovat zamýšlenému použití

Rizika ceny-stupeň nejistoty, že bude dodržen rozpočet

Rizika podpory-stupeň nejistoty, že software půjde snadno opravovat, upravovat a zlepšovat

Rizika času-stupeň nejistoty, že časový harmonogram bude dodržen a že produkt bude dodán včas

Čtyři kategorie dopadu

• **zanedbatelný**

• **kritický**

• **marginální**

• **katastrofický**

Tvorba tabulky rizik

Podle kontrolního seznamu sepište všechna rizika, která připadají pro náš projekt v úvahu.

Uvedte jejich kategorii (velikost softwaru, obchodní, zákaznická, technická,...).

Pro každé riziko určete pravděpodobnost, že nastane (Odhad pravděpodobnosti jako průměr z odhadů každého člena týmu.).

Pro každé riziko určete jeho dopad D (1-4) pro případ, že nastane.

Tabulku rizik uspořádejte podle hodnoty pravděpodobnosti a dopadu.

Vedoucí projektu stanoví dělicí čáru.

Nad ní jsou významná rizika, kterým je třeba se věnovat, rizika pod ní mohou být znovu hodnocena a přerovnána podle sekundárních priorit.

Významná rizika jsou rizika s velkým dopadem a průměrnou nebo vysokou pravděpodobností a případně s nízkým dopadem ale vysokou pravděpodobností.

Pro všechna rizika nad dělicí čarou se sestaví plán RMMM (Risk Mitigation, Monitoring and Management)

- plán zmírnění, monitorování a řízení rizik.

RMMM = Risk Mitigation, Monitoring and Management = zmírnění, sledování a řízení rizik

• snaha vyhnout se rizikům, nebo je zmírnit

• monitorovat rizika během projektu

• řídit průběh rizikových situací podle připraveného plánu

Zvážit cenu RMMM a její užitek.

Plán RMMM

I. Úvod 1. Obsah a účel dokumentu

2. Přehled hlavních rizik

3. Odpovědnosti

a. Management

b. Technický personál

II. Tabulka rizik projektu

1. Popis všech rizik nad dělicí čarou

2. Faktory ovlivňující pravděpodobnost a dopad

III. Zmírnění, monitorování a řízení rizik

N-té riziko

a. Zmírnění

i. Obecná strategie

ii. Specifické kroky k zmírnění rizika

b. Monitorování

Faktory, které mají být monitorovány

Způsob monitorování

c. Management

Plán pro případ, že nastane riziková situace

Speciální zřetel

IV. Harmonogram iterací plánu RMMM

V. Závěr

Smysl monitorování rizik

- odhadnout, zda předpokládaná rizika se mohou ve skutečnosti objevit
- zajistit, aby se kroky plánované proti rizikům, řádně použily
- sbírat informace, které mohou být využity pro příští analýzu rizik

Odezvy na rizika = Opatření, která mají sloužit jako odezvy na vybraná rizika.

Mohou spadat do jedné ze tří kategorií:

Předcházení: Navržené opatření by mělo vyloučit danou hrozbu, obvykle eliminováním její příčiny.

Zmírnění: Navržené opatření má snížit dopad nebo pravděpodobnost rizika.

Akceptace: Navržené opatření se provede v případě, že riziko nastane.

Navržená opatření mohou být následujících typů:

Havarijní plány: Jsou to plány ošetření rizikových událostí, sestávající se z akčních kroků, která je nutno přijmout, jestliže riziková událost nastane.

Alternativní strategie: Rizikovým událostem lze někdy předejít změnou-li strategii řešení.

Rezervy: Jsou to opatření, která mají zmírnit rizika nákladů a harmonogramu. Často jsou blíže specifikována účelem, to znamená, jaká rizika mají být zmírněna (např. provozní rezerva, rezerva na nepředvídané události, časová rezerva).

Obstarávání: Některé činnosti mohou být méně rizikové, budou-li provedeny externí firmou, která má s nimi větší zkušenosti.

Pojištění: Pojistné smlouvy mohou odstranit nebo zmírnit některá rizika.

Testování softwaru

Testování má zvýšit pravděpodobnost, že v programu nejsou chyby.

Strategie testování integruje metody návrhu testů do celkového plánu procesu tvorby softwarového produktu.

Strategie testování zahrnuje plánování testů, návrh testů, provedení testů a vyhodnocení jejich výsledků.

- Testování začíná na úrovni modulu a pokračuje "směrem ven" k integraci kompletního počítačového systému.
- Pro různé situace se používají různé techniky testování.
- Pro malé projekty je testování řízeno realizátorem, pro větší nezávislou skupinou.
- Testování a odstraňování chyb jsou rozdílné aktivity, avšak odstraňování chyb musí být zahrnuto do strategie testování.

V&V (verification and validation)

V&V jsou součástí SQA (SW Quality Acquisition).

Verifikace: Ověření, že software správně implementoval specifické funkce.

"Vytvořili jsme produkt správně?"

Validace: Ověření, že software odpovídá požadavkům zákazníka.

"Vytvořili jsme správný produkt?"

Nesprávné názory na strategii testování

Kdo tu práci požaduje, kdo ji bude užívat, jaký bude ekonomický užitek při úspěšném ukončení, je ještě jiná možnost, jak to vyřešit?

- Tvůrce by neměl provádět žádné testování!
- Software má testovat někdo nezávislý, který to provede bez jakýchkoli ohledů
- Tester má vstoupit do projektu až v okamžiku testování.

Tvůrce by měl testovat jednotky (moduly), v mnoha případech řídí také integrační testy.

Až v okamžiku, kdy je software kompletní nastupuje nezávislá skupina testérů (ITG - independent test group).

Systémové požadavky

analýza požadavků - návrh
- kód

jednotkový test = *white box testing*

integrační test = *verifikace programové konstrukce*

validační test = *ověření, že program vyhovuje požadavkům na funkci, chování a provedení (black-box).*

systémový test = *test v kombinaci s ostatními systémovými prvky - HW, databáze, uživatelé.
(black-box částečně white-box -pokrytí hlavních cest řízení)*

Kdy skončit testování - jak poznat, že jsme testovali dostatečně?

Musa, J.D., Ackerman, A.F.: *Qualifying Software Validation: When to Stop Testing?* IEEE Software, May 1989, pp. 19-27

Logarithmic Poisson execution-time model: $f(t) = (1/p) \ln (I_0 p t + 1)$

kde $f(t)$ je kumulativní počet poruch, které se očekávají, že nastanou po testování softwaru po určitou dobu t ,

I_0 je počáteční softwarová **intenzita poruch** (poruchy za časovou jednotku) na začátku testování
 p exponenciální snížení intenzity poruch, po odhalení chyb a jejich odstranění

Okamžitá intenzita poruch $I(t)$: $I(t) = I_0 / (I_0 p t + 1)$

Testování jednotek

Testování zaměřené na verifikaci malých jednotek softwarového návrhu - modulů. Podle popisu návrhu procedur jsou testovány důležité cesty uvnitř modulu. Testování jednotek je prováděno metodami white-box testing, paralelně pro více modulů.

V rámci testování jednotky se prověřuje

- rozhraní
- lokální datové struktury (zda dočasně uložená data zachovávají svou integritu)
- okrajové podmínky (zda modul pracuje správně na hranicích, omezujících výpočet)
- nezávislé cesty (zaručující, že každý příkaz bude proveden alespoň jednou)
- cesty pro zpracování chyb

Pokud modul ovládá externí I/O, musí být doplněny další testy:

- Správné atributy souborů?
- Odpovídá příkaz I/O specifikacím?
- Jsou soubory před použitím otevírány?
- Ošetření I/O chyb?
- Správné příkazy OPEN/CLOSE?
- Odpovídá velikost bufferu velikosti záznamu?
- Správné podmínky EOF?
- Kontrola textových výstupních informací?

Pro lokální datové struktury

- Nevhodné a nekonsistentní typy?
- Nesprávné (překlepy) jména proměnných?
- Podtečení, přetečení a adresní výjimky?
- Chybná inicializace nebo default hodnota?
- Nekonsistentní datové typy?

Kromě toho je potřeba kontrolovat zpracování globálních dat.

Mezi časté chyby patří

- nepochopené nebo nesprávné priority aritmetických operací
- smíšené operace (mixed mode)
- nesprávná symbolická reprezentace výrazu
- nesprávná inicializace

Antibugging

Předvídání chybného chování při návrhu cesty pro zpracování chybného stavu, které ukončují výpočet s chybovým hlášením. Je třeba testovat, zda nenastane některý z následujících případů:

- Popis chyb není příliš inteligentní.
- Oznamené chyby nekorespondují se skutečnými.
- Chyba způsobí zásah do systému dříve, než je ošetřena.
- Nesprávné ošetření výjimečných podmínek.
- Popis chyby neposkytuje dostatečné informace k lokalizaci chyby.

Pomocné programy pro testování modulů

řídící programy - drivers (nahrazují nadřazené programy)

testovací kódy (makety) - stubs (nahrazují volané podprogramy).

Někdy nelze tyto programy napsat pro každý modul, pak se musí testování jednotky odložit na úroveň kroku integračního testování.

Integrační testování

- Přístup "velkého třesku"

- Inkrementální integrace

Integrace shora-dolů a zdola-nahoru

Integrace shora-dolů

začíná se hlavním řídicím modulem a postupuje se směrem dolů: buď strategií do hloubky nebo do šířky.

- Hlavní modul je "driver", "stubs" nahrazují všechny podřízené moduly
- Postupně (buď podle přístupu do hloubky nebo do šířky) jsou nahrazovány "stubs" skutečnými moduly.
- Po každém přidání modulu je systém otestován.
- Regresní testování má zajistit, že nebyly zavlečeny nové chyby.

Problém může nastat, když očekáváme významné výstupy z podřízených modulů, které nám "stubs" neumí poskytnout.

Úrovně složitosti "stubs":

- zobrazí zprávu
- zobrazí aktuální parametry
- navrácí hodnoty z tabulky nebo externího souboru
- provádí vyhledání z tabulky podle vstupních parametrů a vrací příslušné výstupní parametry

Integrace zdola nahoru

Eliminuje se potřeba "stubs".

- Nejnižší moduly jsou spojeny do skupin (clusterů), které provádí specifické funkce.
- Je napsán "driver", který je koordinuje.
- Skupina (cluster) je otestována.
- Drivery se odstraní a skupiny se pospojují směrem výše v programové struktuře.

Regresní testování

Testuje se, zda nenastal vedlejší efekt přidáním nového modulu (zavlečené chyby) - opakováním předchozích testů.

Existují nástroje na toto testování (Capture-playback tools) - opakují tři rozdílné třídy testů:

- reprezentativní vzorek testů, zkoušejících všechny softwarové funkce
- testy zaměřené na funkce, které by mohly být pravděpodobně zasaženy změnou
- testy softwarových komponent, které byly změněny

Validační testování

Provádí se po integraci a slouží k ověření, že software splňuje "rozumná očekávání" zákazníka, která jsou definovaná ve specifikacích softwarových požadavků ("validační kritéria").

Provádí se metodami black-box testing.

Pokud je sw určen pro jednoho uživatele, předá se mu do užívání k **akceptačním testům**.

Pokud je SW určen pro více různých uživatelů, provádí se alfa a beta testování.

Alfa testování provádí zákazník v řízeném prostředí dodavatele ("programátor se mu dívá přes rameno").

Beta testování se provádí u jednoho nebo více zákazníků. Vývojář není přítomen. SW se zkouší v "živých" podmínkách. Zákazník zapisuje všechny problémy (skutečné i imaginární) a určitých intervalech je posílá dodavateli.

Systémové testování

Je série různých testů, která prověřuje celý počítačový systém (HW, SW prostředí, databáze, lidi...).

Recovery testing (testování obnovy)

Systémové testování ověřující, že poruchy byly řádně ošetřeny v předepsaném čase.

Pokud je ošetření automatické, vyhodnocuje se re-inicializace, mechanismus checkpointů, obnova dat, restart. Je-li prováděno manuálně, ověřuje se, zda "mean-time to repair" byl v limitu.

Security testing (bezpečnostní testování)

Testování odolnosti proti útokům hackerů.

Tester supluje roli hackera a snaží se proniknout do systému. Má-li dost času a zdrojů, tak se mu to podaří. Úlohou systémového návrháře je, aby cena proniknutí do systému byla větší než cena informací, které lze získat.

Stress testing

Cílem je prověřit program v abnormální situaci (kvantita, frekvence nebo obsah).

Jak dlouho mohu systém namáhat, než padne?

Např.

- generovat 10 přerušení za sekundu
 - vstupní data zvětšit o řád než je dovoleno a vyhodnotit chování vstupní funkce
 - požadovat maximální paměť či jiné zdroje
 - pokusit se způsobit problém v operačním systému
 - velké nároky na disk
- apod.*

Variantou je **sensitivity testing**, které se snaží objevit kombinace dat (v rámci platného omezení), které mohou způsobit nestabilitu či nesprávnou funkci.

Performance testing (pro zapouzdřené s. nebo s. reálného času).

Ladění (The art of debugging)

Zatímco testování lze plánovat a systematicky provádět podle nějaké strategie, ladění, které je důsledkem úspěšného testování (našly se chyby) je více méně uměním.

Oprava nesouhlasu mezi očekávaným a skutečným výstupem spočívá v nalezení příčiny daného symptomu.

Příčina se buď najde a chyba se opraví, nebo se musí navrhnout jiný test, který ji pomůže lokalizovat

Zatímco testování lze plánovat a systematicky provádět podle nějaké strategie, ladění, které je důsledkem úspěšného testování (našly se chyby) je více méně uměním.

Oprava nesouhlasu mezi očekávaným a skutečným výstupem spočívá v nalezení příčiny daného symptomu.

Příčina se buď najde a chyba se opraví, nebo se musí navrhnout jiný test, který ji pomůže lokalizovat

Všechny přístupy mohou být podpořeny ladicími nástroji (debugging tools) - traces, automatické generátory testů, výpisy paměti, křížové odkazy (cross-reference), ladicí nástroje překladačů.

Často stačí vysvětlit problém jinému kolegovi (jiný pohled) a chyba se objeví.

Techniky testování softwaru

Charakteristika testování

- Testování je proces spouštění programu za účelem nalezení chyb.
- Dobrá testovací data jsou taková, která s velkou pravděpodobností objeví dosud neobjevené chyby.
- Úspěšný test je takový, který objevil dosud neobjevené chyby.

Principy testování

- Testy by se měly vztahovat k požadavkům zákazníka
- Testy by měly být plánovány v předstihu
- *Princip Pareto (pravidlo 80:20)* : většina všech neobjevených chyb má původ v několika málo modulech
- Testování by mělo začít testováním "v malém" a pokračovat testováním "ve velkém".
- Úplné otestování není možné
- Testování by mělo být vedeno nezávislou třetí stranou

Testovatelnost

Výčet vlastností vedoucích k dobré testovatelnosti:

Spustitelnost = čím lépe SW pracuje, tím účinněji může být otestován
chyby nebrání běhu programu, může být testován a vyvíjen současně

Přehlednost = testuješ, co vidíš

různé výstupy pro různé vstupy, stavy systému a proměnné jsou viditelné během chodu, faktory ovlivňující výstup jsou viditelné, nesprávné výstupy jsou lehce identifikovatelné, vnitřní chyby jsou automaticky detekovány a zaznamenávány, je dostupný zdrojový kód

Kontrolovatelnost = čím lépe lze software kontrolovat/řídít, tím spíše lze testování automatizovat a optimalizovat

Řízení kvality SW

Plánování jakosti spočívá ve stanovení cílů jakosti a specifikování procesů pro splnění těchto cílů. Výsledkem této činnosti je **plán řízení jakosti**.

Řízení jakosti je zaměřené na splnění požadavků na jakost. Spočívá ve sledování konkrétních výsledků projektu a posuzování, zda odpovídají standardům a stanoveným požadavkům. Pokud neodpovídají, navrhuje se způsoby odstraňování příčin nevyhovujícího plnění.

Zabezpečování jakosti je zaměřené na poskytování důvěry, že jsou splněny požadavky na jakost. Je tedy zaměřeno na to, aby zákazník, vedení prováděcí organizace, případně další strany nabyly důvěry v kvalitu procesu a produktu (*je to vlastně ujištění zákazníka nebo třetí strany o kvalitě*). Činnosti pro zabezpečování jakosti mohou být prováděny i externě, třeba formou externího auditu.

Harmonogram řízení jakosti

Začátek projektu: 28.2.2006 Konec projektu: srpen 2006

ID	Jméno	Čas	Začátek	Konec	Závislosti
A1	Zpracování úvodní studie	14 dní	7.3.	21.4.	
A2	Kontrola úvodní studie	1 den	22.4.	22.4.	A1
B1	Zpracování analýzy	26 dní	23.3.	18.4.	A
B2	Kontrola analýzy	1 den	19.4.	19.4.	B1
C1	Zpracování návrhu	11 dní	2.5.	12.5.	B
C2	Kontrola návrhu	1 den	15.5.	15.5.	C1
D1	Implementace databáze				C
D2	Kontrola implementace databáze (White-Box, Black-Box)	3 dny			D1
D3	Implementace autorizačního a aplikačního serveru				D1
D4	Kontrola implementace autorizačního a aplikačního serveru (White-Box, Black-Box)	5 dní			D3
E1	Knihovny a vnější kostra klienta				C
E2	Kontrola jednotlivých knihoven (White-Box, Black-Box)	5 dnů			E1
E3	Kontrola kostry (White-Box, Black-Box)	1 den			E1
F1	Implementace jednotlivých formulářů				D, E
F2	Kontrola jednotlivých formulářů (White-Box, Black-Box)	14 dní			F1
G1	Integrace, shora dolů				D, E, F
G2	Integrační testy (Black-Box)	3 dny			G1
H	Systémové testy	14 dní			G
H1	Testy zotavení	4 dny			
H2	Testy bezpečnosti	7 dní			
H3	Zátěžové testy	3 dny			
I	Validační testy	30 dní			H
I1	Alfa testing	5 dní			
I2	Beta testing	25 dní			
J1	Zpracování uživatelské dokumentace	30 dní			C
J2	Kontrola uživatelské dokumentace	10 dní			J1

Legenda

A2

Evaluace úvodní studie, kontrola plnění požadavků, kontrola integrity dokumentů

B2

- C2** Evaluace analýzy, kontrola plnění požadavků, kontrola integrity analýzy, verifikace s úvodní studií
- D2** Evaluace návrhu, kontrola plnění požadavků, kontrola integrity návrhu, verifikace s analýzou
- D4** Evaluace implementace databáze (kontrola návrhu tabulek)
Verifikace s návrhem
Databáze zajišťuje konzistenci - obsahuje uložené procedury a triggery, validace tohoto kódu metodou White i Black Box
- E2** Validace kódu autorizačního a aplikačního serveru
Kontrola dodržování autorizační politiky
- E3** Validace kódu pomocných knihoven (SQL, RSA, práce s řetězci, GUI)
- F2** Validace kódu hlavní kostry programu s využitím stubs na místech jednotlivých formulářů
- G2** Validace kódu jednotlivých formulářů, kontrola plnění požadavků, verifikace s návrhem, testování GUI
- H1** Kontrola integrace formulářů do kostry metodou shora dolů, nyní je díky existenci větších funkčních celků možné intenzivnější Black Box testování, testování GUI
- H2** Ošetření a zotavení se z poruch
- H3** Testy odolnosti proti pokusům o zneužití programu
- I1** Testy reakcí na nadměrnou uměle generovanou zátěž (s ohledem na charakter projektu zejména frekvenční)
- I2** První nasazení produktu v praxi s účastí zástupců vývojového týmu
- J2** Nasazení produktu v praxi, podpora zákazníkům pro hlášení chyb (instrukce, jak chyby hlásit; formulář)
Prověření obsahu (jednotnost pojmů, srozumitelnost, věcnost a správnost), formy
Testy, zda postupy uvedené v dokumentaci skutečně vedou k očekávaným výsledkům.