RSP

Klasické a agilní metodiky vývoje software, sestavení a sledování projektového plánu, metody odhadování pracnosti

# Základní koncepty a pojmy z projektového řízení:

* **Definice projektu**: Projekt je časově ohraničená posloupnost unikátních, komplexních a propojených činností, které mají jediný cíl nebo účel, a které musí být dokončeny v určitou dobu, v rámci rozpočtu a podle specifikace.
* **Scope**: Seznam veškerých požadavků a výstupů projektů, který by dodavatel měl vytvořit
* **Mitigační akce**: Aktivita, která snižuje rizika
* **Kritická cesta:** je definována jako (časově) nejdelší možná cesta z počátečního bodu grafu do koncového bodu grafu. Každá kritická cesta se skládá ze seznamu činností, na které by se měl manažer projektu nejvíce zaměřit, pokud chce zabezpečit včasné dokončení projektu.
* **Orgchart projektu (investor, PM, TL, podtýmy):** organizační struktura projektu
* **Síťový graf:** je matematickým modelem projektu, který přesně popisuje závislosti jednotlivých činností. Jde o orientovaný graf, který se skládá z uzlů a hran.
* **Proof-of-concept(PoC):** realization of a certain method or idea in order to demonstrate its feasibility
* **Další**: Výstupy projektu (deliverables), Milníky, Aktivity (tasks), Odhadování aktivit, Verifikace odhadů, Rezerva, Zdroje, Rizika, Odhad závažnosti rizika (dopad, pravděpodobnost výskytu), Issues, Projektový plán, Prototypování, Feasibility Study

## Validace vs. verifikace

* **Verifikace:** je kontrola, zda vyvíjený systém vyhovuje specifikacím (našemu návrhu).

Mezi metody verifikace patří většina testování, kterého se neúčastní zákazník (jednotkové testy, integrační testy apod.), inspekce kódu, simulace či různé formální analýzy a verifikace.

* **Validace:** je kontrola, zda vyvíjený systém splňuje to, co od něj zákazník očekává.

Mezi metody validace patří např. akceptační testy, které se provádí u zákazníka

## Severita vs. Priorita

* **Severita**: závažnost chyby z hlediska fungování systému
* **Priorita**: Míra urgence opravy chyby

## Entry exit kritéria

* **Entry kritéria**: Sada podmínek, které musí být splněny, aby konkrétní aktivita mohla začít. (podobně jako definition of ready u user story)
* **Exit kritéria**: Sada podmínek a kritérií, dohodnutých s investorem a managementem projektu, které musí být splněny, aby byla konkrétní aktivita považována za splněnou. (podobně jako definition of done u user story)

# Plánování

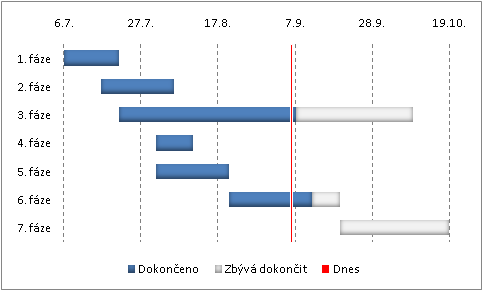
**Co musí projektový manažer zajistit**

* Plánování
* Organizace týmu
* Měření postupu
* Sledování rizik

**Kroky vedoucí k vytvoření plánu**

* Určení cíle projektu
* Určení artefaktů (deliverables), které musí být dodány
* Určení úkolů vedoucích k vytvoření těchto artefaktů
* Work breakdown structure(WBS)
* Identifikace vztahů mezi těmito úkoly
* Identifikace možných rizik průběžně
* Odhadování pracnosti více fází
* Naplánování zdrojů na projekt
* Vytvoření výchozího projektového plánu – více fází

**Gantův diagram**



**WBS** (**Work Breakdown Structure**), překládá se jako rozpad, rozpis práce nebo jako osnova rozpisu práce, často se používá zkratka WBS. Jedná se o jednoduchou [analytickou techniku](https://managementmania.com/cs/analyzy-analyticke-techniky), jejímž cílem je rozložit [projekt](https://managementmania.com/cs/projekt) na jednotlivé činnosti až do takové úrovně podrobnosti, aby k nim bylo možné přiřadit odpovědnosti, pracnost a časový horizont.

# Řízení rizik / změnové řízení

**Negativní události pro projekt**:

* + změna scope
  + změna milníků
  + výpadky zdrojů
  + podceněné odhady pracnosti
* Negativní dopady
  + Protažení projektu v čase
  + Prodražení projektu
  + Selhání projektu (nedodá očekávaný výsledek)
  + Plýtvání zdroji na projekt
  + Dodávka v nižší kvalitě, než je očekávání
  + Ztráta reputace realizačního týmu
  + Ztráta důvěry zadavatele a dalších zakázek
  + Soudní spory
* 40% -70% chyb má původní příčinu v špatně specifikovaných požadavcích

## Rizika

Riziko je nejistá událost nebo soubor událostí, které pokud nastanou, budou mít dopad na dosažení cílů projektu

* Ocenění rizik: Cena rizika = Pravděpodobnost × Dopad

**Projektová rizika**

* Rizika, která mohou mít dopad na dodávku výstupů projektu v plánovaném čase, za plánovanou cenu či v plánovaném rozsahu
* Projektová rizika ovlivňují: čas, scope, cenu, kvalitu výsledného SW

**Produktová rizika**

* Rizika, která ovlivňují některou z požadovaných či očekávaných vlastností vytvářeného produktu
* Někdy nazývána kvalitativní –mají dopad na výslednou kvalitu produktu
* Produktová rizika ovlivňují: kvalitu výsledného SW

### Identifikace rizik

Snažíme se zjistit všechna relevantní rizika: Vhodný je brainstorming se členy týmu. Pak seznam rizik profiltrovat

Rizika je nutné rozpadnou na dílčí problémy. **Hierarchický rozpad rizika**:

* Tým: Nedostatečná znalost, Nedostatek motivace, Nedostupnost lidí
* Technologie: Nedostatečný výkon, Selhání zálohování

**Diskuze**

* Investor, project manager (PM), jeho team leaders, business PM na straně zákazníka, stakeholders

**Brainstorming**

* Méně formální skupinka, např. 5-10 lidí
* Zapisujte všechny nápady i méně relevantní

**Kontrolní seznamy (checklists)**

* součást projektových metodik
* často jsou zobecněné, přizpůsobte pro konkrétní projekt
* nepředkládejte lidem, od kterých chcete vědět rizika, omezí to jejich kreativitu

**Lessons learned**

* možný problém: officiální lessons learned mohou být "filtrované" a nepopisují skutečné příčiny problémů
* lepší mohou být neoficiální zkušenosti členů předchozích projektů

**Seznamy rizik (risk logs) z předchozích projektů**

## Co evidujeme u rizik:

* Kategorie rizika
* Název rizika
* Popis rizika
* Pravděpodobnost
* Dopad
* Popis aktivit pro řízení rizika (mitigační akce)
* Status rizika (Open/Closed)
* Vlastník rizika
* Osoba zodpovědná za realizaci naplánovaných opatření

## Opatření pro snížení rizika

* **Avoid**: přijmeme opatření snižující pravděpodobnost rizika na nulu
* **Accept**: přijmeme fakt, že určité riziko existuje, ale neděláme nic pro jeho zmírnění
* **Reduce**: aktivním opatřením pravděpodobnost nebo dopad snižujeme
* **Fallback**: nesnižujeme pravděpodobnost výskytu rizika, ale zmírňujeme dopady tím, že máme připraven záložní plán
* **Share**: nesnižujeme pravděpodobnost rizika, ale snižuje její dopady rozkladem na více subjektů, typicky mezi dodavatele a odběratele
* **Transfer**: riziko můžeme přenést na třetí stranu: pojištění odpovědnosti za škodu, přenos na subdodavatele v rámci smluvního vztahu atd.

# Možnosti řízení projektů

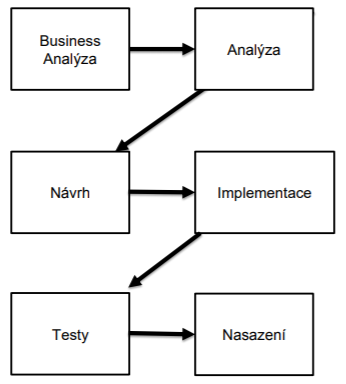
Řízení projektu jsou spojité nádoby:

* KKTR: Kvalita, Kvantita, Termíny, Rozpočet
* KKTRM: Kvalita, Kvantita, Termíny, Rozpočet, Morálka

**Základní fáze:** Zahájení => Plánování => Realizace => Uzavření

**Boehmův první zákon:** Chyby jsou nejčastější ve fázích požadavků a návrhu a čím později odhalíme chybu v průběhu vývoje systému, tím je dražší

## Vodopádový model

**Silné stránky**

* Už od začátku je jasný plán => víme co bude
* snadné řízení zdrojů (paralelismus projektů)
* Není třeba aby tým byl na jednom místě – analýza u nás, implementace v Malajsii
* Snadné předání externímu dodavateli

**Slabé stránky**

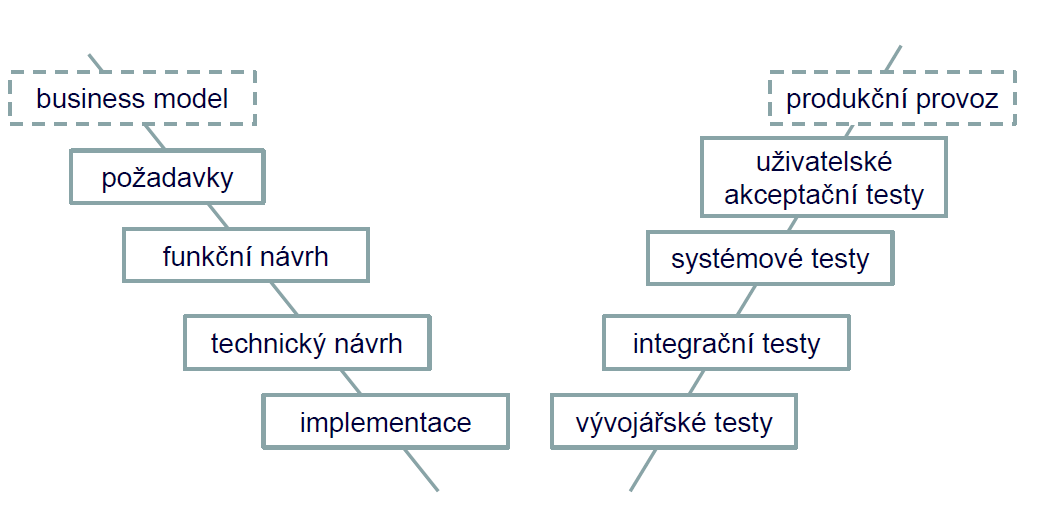
* Špatně reaguje na změnu
* Dlouho trvá, než zákazník vidí nějaký výstup
* Vyžaduje detailní plán a specifikaci – každé přeplánování je drahé
* Není zaměřen na přínos klientovi – je zaměřen na dodání v nějaký čas – dodání podle specifikace není často totožné s tím co zákazníkovi pomůže

**Nevýhody**

* Pro složitější systémy je obtížné dokončit jednu fázi, aby mohla začít další
* Problém s fixací požadavků na začátku a změnové požadavky v průběhu
* "paralysis by analysis" problém
* Změna požadavků v průběhu, potřeba změnit návrh, nárůst ceny
* Skrytá technologická rizika

## V model

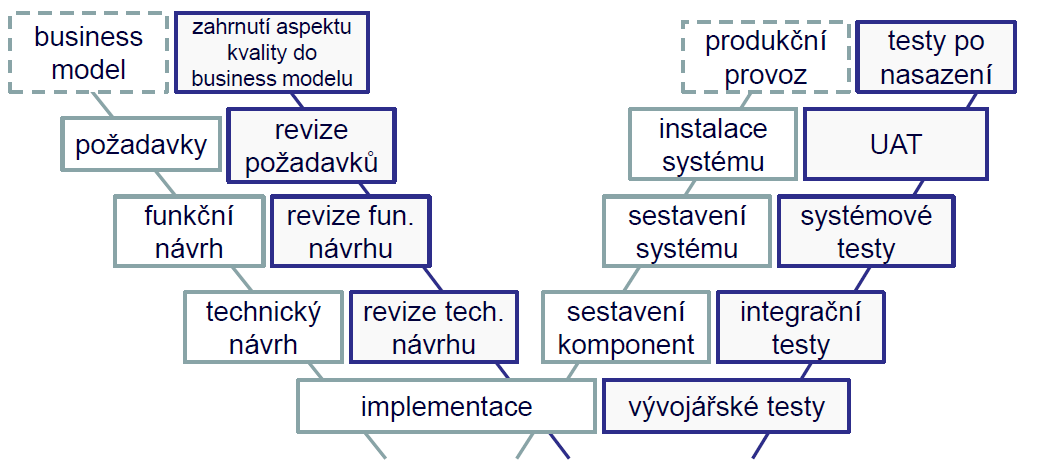
* Podobný jako vodopádový model, ale ke každé fázi přidává způsob testování.
* Větší důraz na testování
* Drahé chyby způsobené špatnými požadavky odhalujeme až ke konci testování!



**čas**

## W model

* Vychází z V modelu (rozšiřuje)
* Každé fázi přípravy systému odpovídá kontrolní fáze
* V praxi levnější než předělávání a opravy na poslední chvíli, ale vyžaduje počáteční investici, proto není příliš rozšířený



**čas**

## Iterační řízení

**Charakter**

* Častý sběr požadavků na změnu
* Tým v jedné lokalitě
* Iterace 2-4 týdny
* Část řešení není známa
  + nejsou detaily -> změny
* Zpětná vazba s klientem
* Jednoduché/lehké zapojení klienta

**Potřeby:** Malý tým**,** Zkušení pracovníci

**Nevýhody:** Zapojení klienta**,** Týmy v jedné lokalitě**,** Není vždy jasné, jak to dopadne

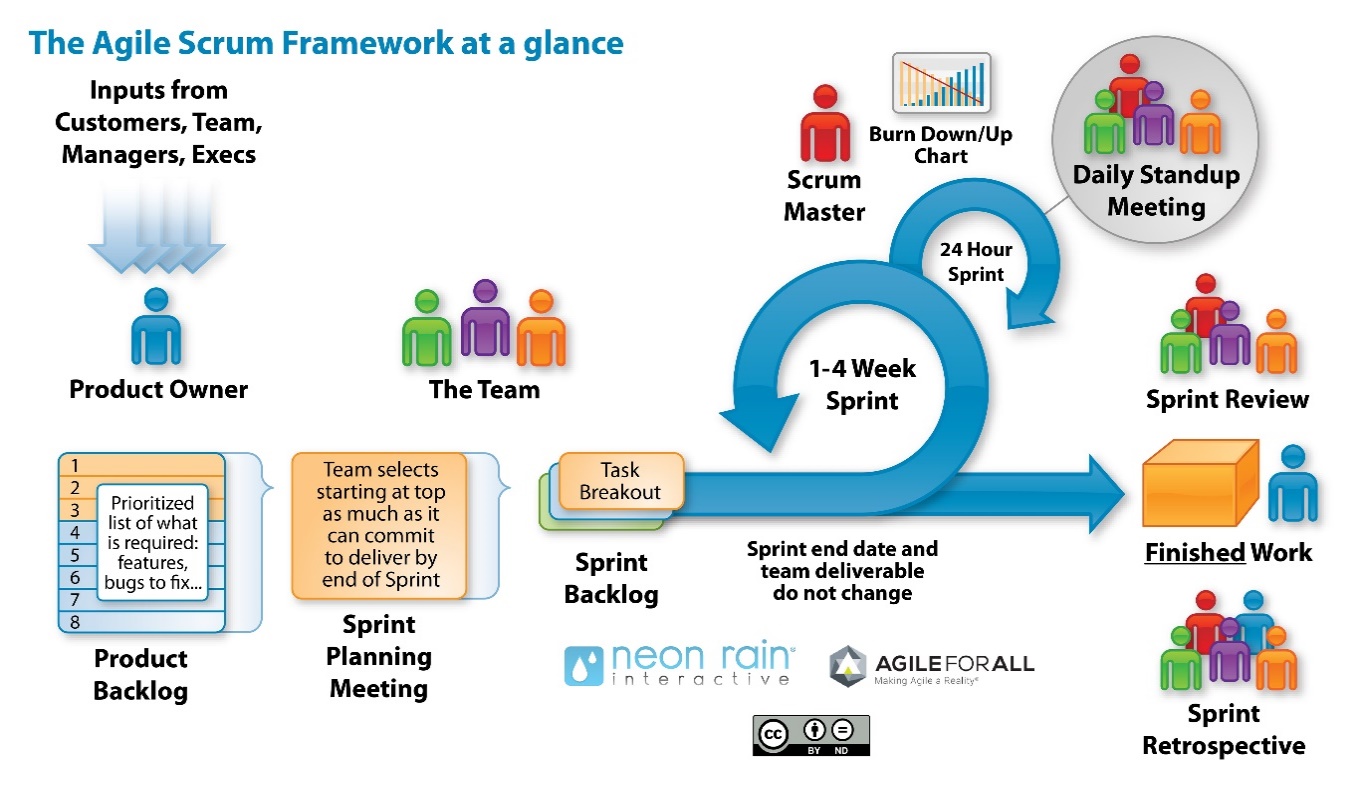
## Agilní přístup

* **Cíl**: Spokojený zákazník, spokojený programátor
* **Kdy**: Zkušení programátoři, Očekává se hodně změn, Malý tým
* **Motto/manifest**:
  + Člověk a komunikace je nad procesy a nástroje
  + Fungující software je nad vyčerpávající dokumentaci
  + Spolupráce se zákazníkem je nad přesně sjednanou smlouvou
  + Reakce na změnu je nad přesné dodržování plánu (Ale zas má agile pravidlo – nezasahovat do iterace. Proto krátké iterace)
* **Kdy ne**: Velké programátorské týmy, Dislokované týmy, Nutnost fixace projektového trojúhelníku
  + Velký tým ve velké kanceláři… nemůže dělat meetingy... Horší komunikace
  + Dislokované týmy – nejsou meetingy
  + Agile – se nedá přikázat, musí se aplikovat všechno - =
  + Kritické systémy – vyžadují dokumentaci, většinou jsou předem jasné požadavky

## Scrum

* iterativní a inkrementální metodologie agilního vývoje softwaru používaná na řízení produktového vývoje.
* zpochybňuje předpoklady "tradičního, sekvenčního přístupu" k vývoji produktu
* Klíčový princip scrumu je jeho pochopení, že během projektu mohou zákazníci změnit názor o tom, co chtějí a potřebují
* Scrum používá empirický přístup, podle kterého problém nelze zcela pochopit nebo definovat a proto se soustředí na maximální schopnost týmu rychle dodat a reagovat na nové požadavky

### Princip



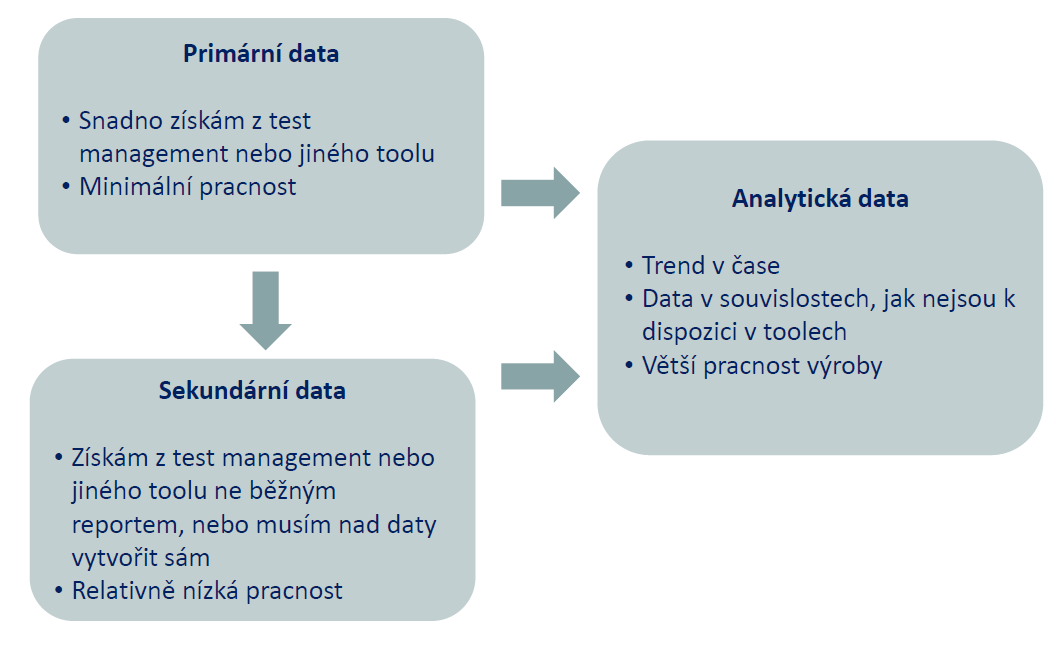
# Metriky pro sledování a zajištění kvality

* Měření efektivity testů (aktuální stav, vyhodnocení zavedené změny)
* Detekce bottlenecků a neefektivit
* Základ pro exit kritéria
* Základ pro akceptační kritéria
* Vstupy do odhadovacích metod
* Měření kvality dodávek dodavatelů
* Reporting

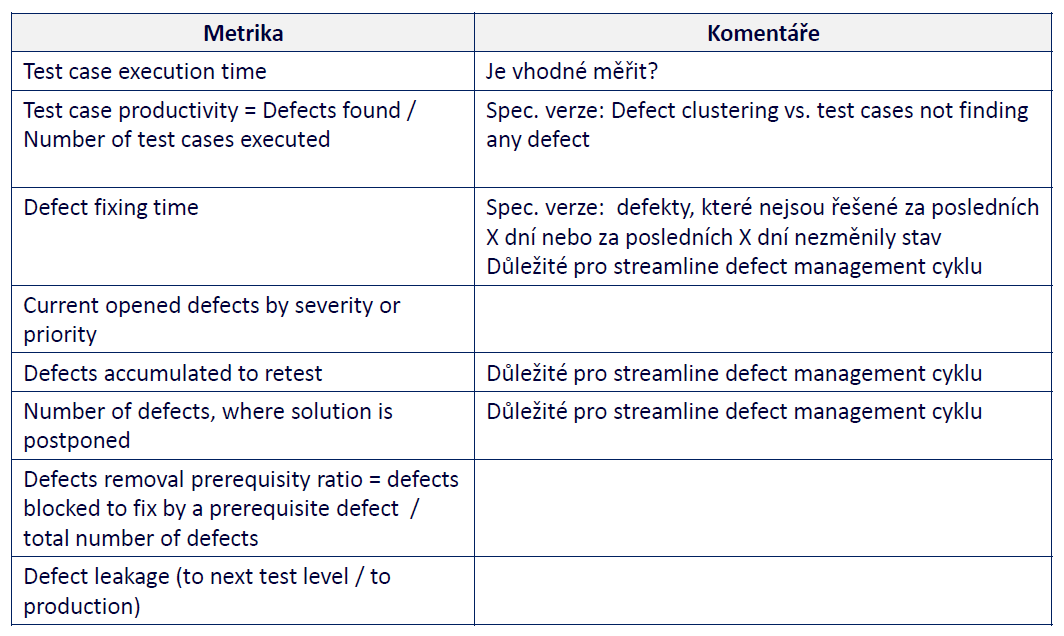
**Zajímavost: Hawthornský efekt:** Efekt popisuje jev, kdy je chování účastníků sociálně-psychologického experimentu vztaženo pouze ke zvláštní navozené situaci a událostem během experimentu. To je nicméně silně ovlivněno vědomím, že jde o experiment, a může tak docházet k významným zkreslením při interpretaci výzkumných zjištění**.**

**=>** sledování zaměstnanců při repetitivně práci zvyšuje výkonost u kreativní práce je tomu naopak.

## Typy metrik



## Příklady metrik



## Metriky zajištění kvality SW

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Metric** | **Definition** | **Purpose** | **How to interpret** |
| **Review Defect Density** | Review Defect Density defines the effectiveness of reviews and the efficiency of writing test scripts. | Use of this metric is to judge the effectiveness and the quality of test script written. To track the efficiency of the tester in writing the test scripts. This metric may also show the reviewer's efficiency. | It’s the number of Review defects found per 1000 Raw Steps. The number indicates the Efficiency of the team in writing the test scripts. Lower number more efficient is the team in writing test script. |
| **Defect Leakage** | Defect Leakage defines the number of defects found in production as a result of miss out from the testing teams while execution. | Use of this metrics is to determine the overall efficiency of the testing teams. It also gives a sense of the stability of the system. | Higher the number lower the testing quality. |
| **Requirement Coverage** | The percentage of requirements mapped to Test scripts against total requirements is called as Requirement coverage. | This metric signifies the effectiveness in terms of requirement coverage | More the percentage better the coverage of requirements. |
| **Quality Index** | This metrics signifies the inherent quality of the product. | Quality index is determinant of quality of product under test. It signifies the product quality in percentage. | High the Quality Index implies higher quality of product under test. |
| **Defect Discovery & Defect Fix rate** | Defect Discovery & Defect Fix rate metrics is used to find out whether the application can be released at a planned release date. In effect, it measures the effectiveness of the development and testing team working together for a targeted release date. | This Metric Plays a major role in determining if the product will be ready to be shipped on time. | If the number is greater than one it implies that all the defects will not be fixed on time, and the project will not meet the deadlines. |
| **"Test execution rate"** | Addresses Test scripts not executed by against Total Planned Test scripts | The purpose of this metric is to track the Test scripts not executed against planned for execution. It also helps in analysing the reasons for the Test Slipage. | Lower the number lower is the variance in actual execution against planned. |
| **% Reuse** | % Reuse metrics defines the scripts that are being reused without change in steps. Test Data change is exception. | This Metric signifies the Percent of scripts that are reused, and analysing the low percentage of reuse if applicable. | Out of the total scripts executed what percentage of scripts are reused |
| **Valid defect ratio** | Percentage of defects accepted as valid as against total defects logged. | This metric signifies the Test team's quality in finding & reporting defects. | Higher the percentage better the Testing Teams quality in reporting defects |
| **Defect Re-open rate** | This metrics defines the total Number of defects Re-opened with the total number of valid defects for the phase. | This Metric signifies the amount of rework required by the testing team in the phase under consideration. | High percentage signifies poor quality of defect fixes or poor quality of defect reporting. |
| **Defect Aging** | Defect ageing defines the average life of the defect opened in a particular phase under execution. | To trace average life of a defect. | Average life of the defect as per the severity. |
| **Root cause** | This signifies the Root cause of the defect appearance in the application. | Finding the root causes helps have a better analysis and understanding of the defect. | The various root causes that can be categorised are as follow. 1. Documentation 2. Coding & Design 3. Environment & setup factors |

# Odhadování pracnosti

* Sofistikovaná metoda s nepřesnými vstupy dá nepřesný výstup
* Verifikujte odhady – ideálně proveďte alespoň 2 nezávislé odhady
* Nezaměňujte s postupem „kolega zreviduje odhad“ – viz anchoring effect dále
* Sběr historických dat a kalibrace metody je účinná pro konkrétní prostředí
* … ale kalibraci pak není vhodné kompletně generalizovat: pozor na přenositelnost metody kalibrované pro jedno prostředí do jiného

## V průběhu projektu: Co může způsobit odchylku od původního odhadu?

* Nesprávná nebo nesprávně použitá metoda odhadu
* Do odhadu jsme nezahrnuli očekávanou kvalitu testovaného systému velké množství defektů
* Změny scope projektu
* Špatná nebo chybějící specifikace
* Pozdní dodávky od analytiků nebo vývojářů
* Testovatelnost dodané aplikace
* Nefunkční prostředí
* Komunikace v týmu

## Co nás ovlivňuje

**Anchoring effect (efekt ukotvení)**

* Při rozhodování nebo usuzování se spoléháme na první informaci, kterou dostaneme
* Výrazně snižuje vhodnost postupu „zkontrolujte něčí odhady“

**Snaha vyhovět očekáváním**

* Prakticky jinak zformulovaný anchoring effect
* Pokud se objektivní výsledek výrazně liší od předchozího očekávání nebo konsenzu, máme přirozenou tendenci ho revidovat a hledat kompromis

**Přesnost odhadu podle úrovně detailu**

* Odhad detailních částí – tendence nadsadit dílčí odhady (zaokrouhlování, přidávání drobné rezervy ke každé z detailních činností)
* High-level odhad – opačný jev, může být tendence k nižšímu odhadu, opomeneme nacenit nějaké činnosti a jejich interakce

**Over-confidence effect (nadsazená důvěra ve vlastní úsudek)**

* Subjektivní důvěra ve vlastní odhady je obvykle vyšší než jejich přesnost

**Nevhodná práce s nejistotou**

* Jakmile oficiálně komunikujeme odhad, začneme ignorovat možnou míru nepřesnosti, která v něm je, místo abychom s ní dále vhodně pracovali.
* Tím se zvyšují skrytá rizika projektu

# Odhad velikosti SW

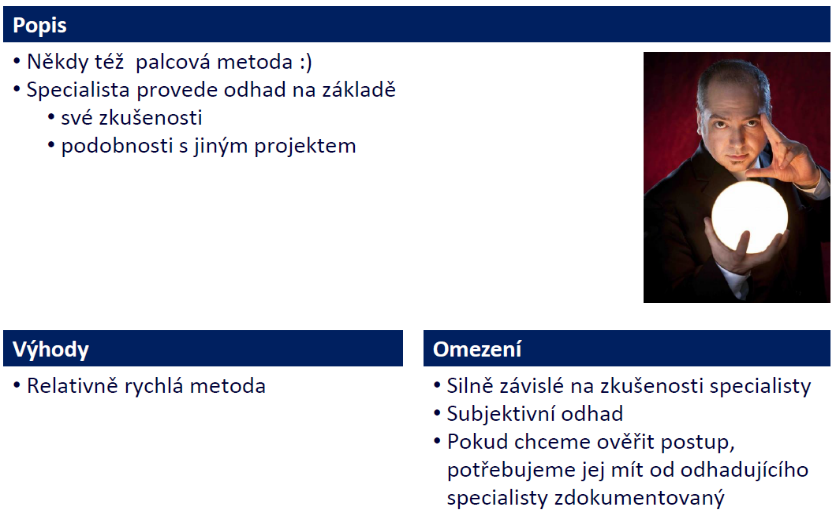
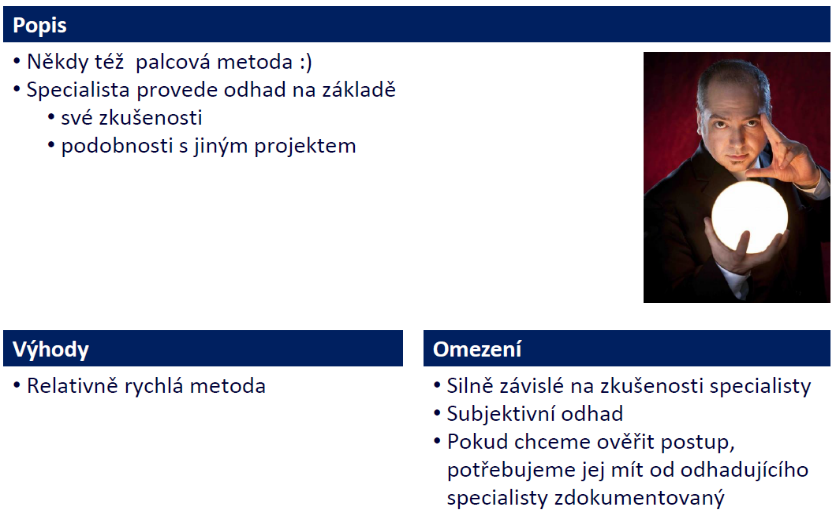
## COCOMO

* Odhad velikosti systému na základě počtu řádků kódu

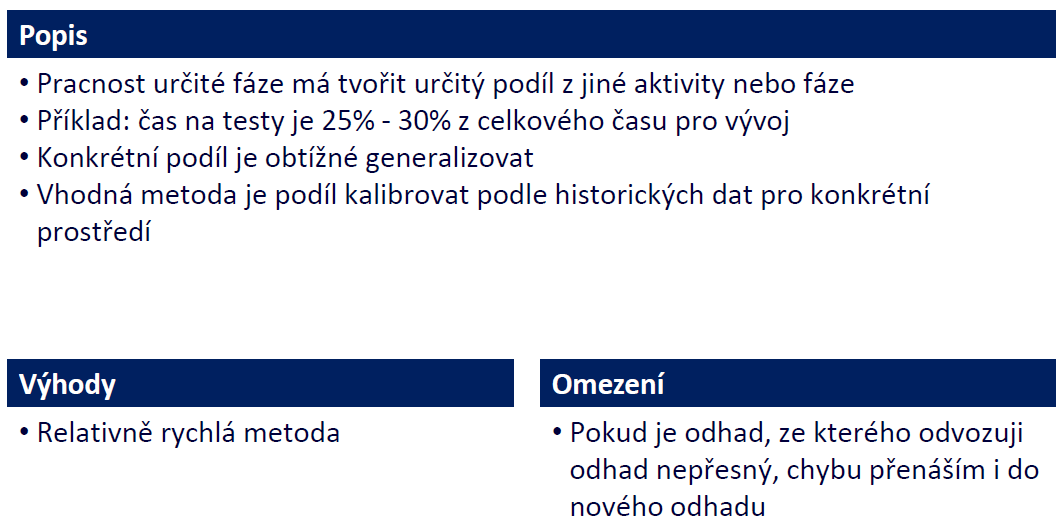
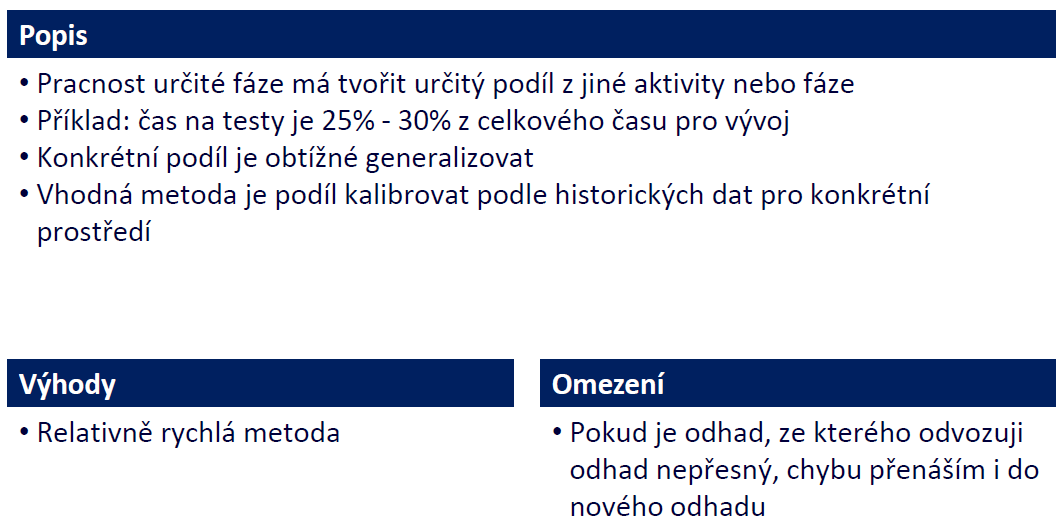
## Function Point

* Odhad podle počtu business usecasů

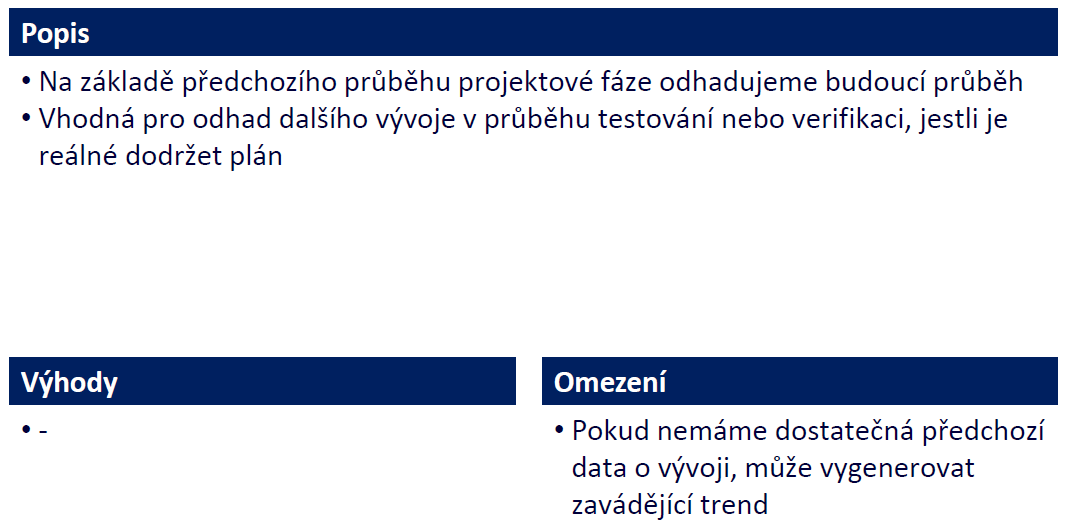
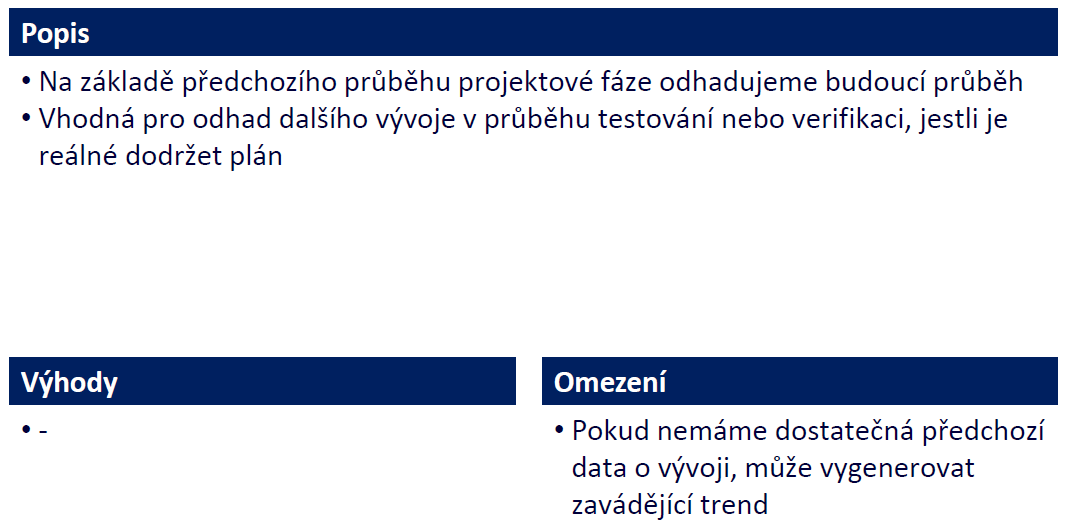
## Expertní odhad



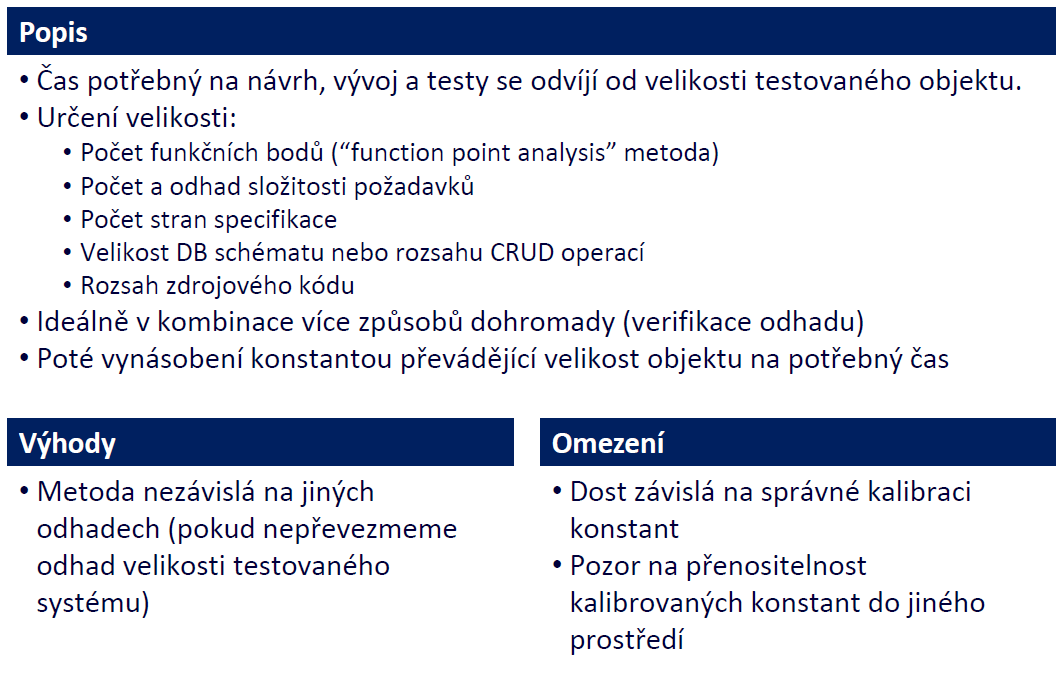
## Podílový odhad:



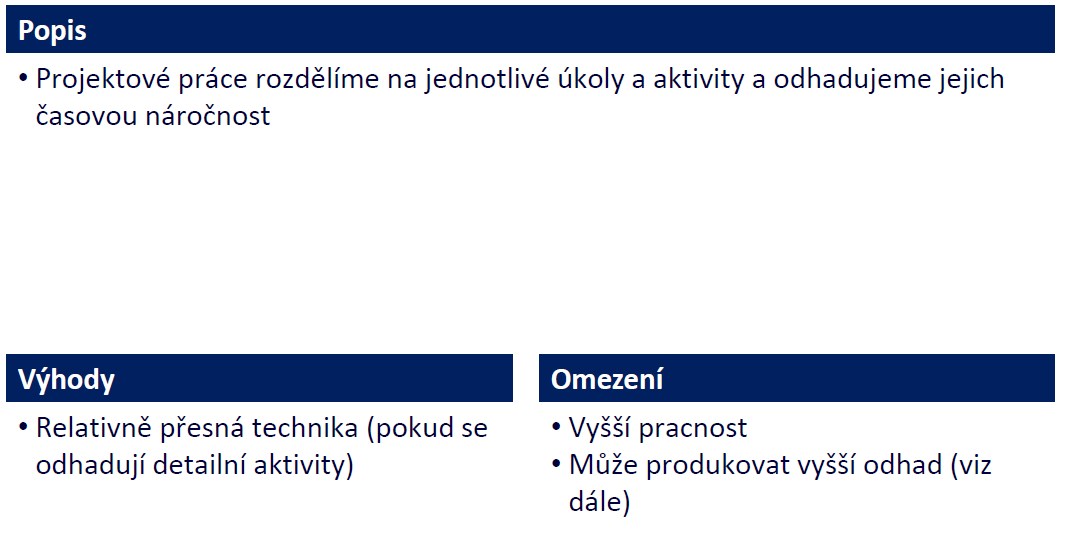
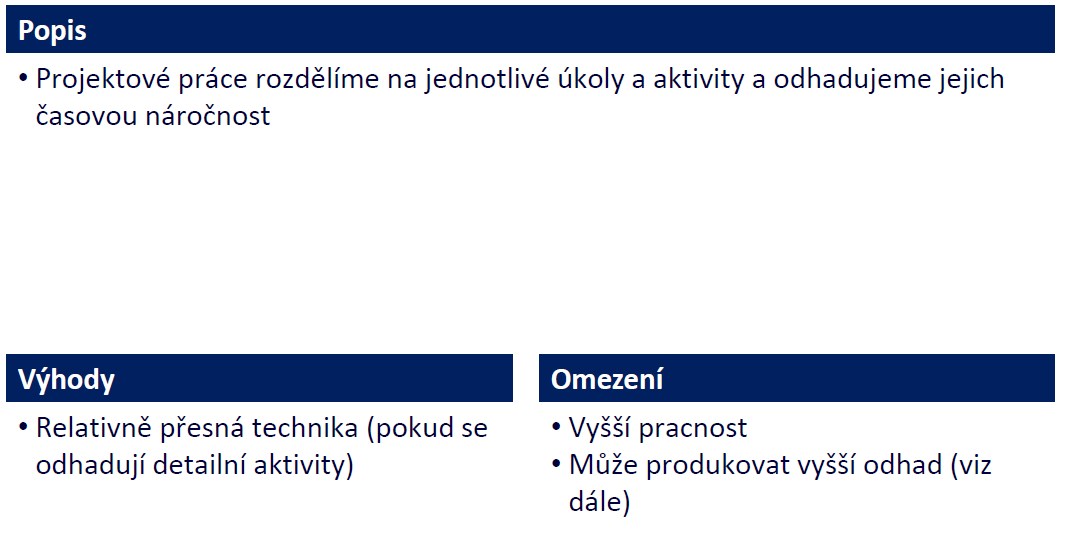
## Extrapolace předchozích dat



## Odhad velikosti vytvářeného systému



## Work Breakdown Structure



# Testovací strategie pro SW

## BDTM: Business driven test managment – z TMAP Next

Business Driven Test Management allows the client to manage testing on the basis of business considerations.

Výstupem je tabulka, která propojuje (odpovídá krokům nahoře):

* **test goals** - získám následující dvě nezávisle a porovnám jejich očekávání
  + overall test goals - od investora nebo managera
  + departmental test goals - od koncových uživatelů
* **evidence požadavků**, částí specifikace, částí testovaného systému
  + od analytiků sesbíráme požadavky na systém a jeho funkce
  + požadavky rozdělíme do kategorií (quality characteristic)
  + propojíme s předchozím, pro každý požadavek by měl být jeden test goal a obráceně
* **analýza rizik**
  + určení priorit, ty plynou z rizik = metoda PRA = product risk analysis
  + určení třídy rizika – pravděpodobnost x dopad
  + kontrola: odpovídají rizikové věci tomu, co se má testovat podle test goals.
* **definované test levels**
  + revize / vývojářské testy / systémové testy / UAT / test v produkci
  + poté se upraví podle aspektů (jestli jsou peníze, čas, zdroje...)
* **druhy testů a jejich intenzita**
  + tabulka test levels X část systému (funkce)
  + u každé dvojice řekneme, jestli nízká, střední, vysoká intenzita
  + znovu úprava podle aspektů

Testovací strategii postupně sestavujeme pomocí následujících kroků:

* Určení cílů testování v součinnosti se zadavatelem a zástupci klíčových uživatelů systému.
* Vytvoření přehledu jednotlivých částí testovaného systému.
* Určení priority testování na základě produktových rizik.
* Určení vhodných úrovní testování (test levels).
* Určení intenzity testování pro jednotlivé části testovaného systému.
* Přiřazení test design technik odpovídající intenzitě testování.
* Odhady pracnosti pro návrh testovacích scénářů a vykonávání testů.
* Revize celkové pracnosti a případná úprava testovací strategie.
* Rozvedení testovací strategie do detailního projektového plánu testů.