Metody řízení projektů – cesta k efektivitě a úspěchu

Ing. Jaroslav Lepka

26. února 2010

Tato prezentace je spolufinancována Evropským sociálním fondem a státním rozpočtem České republiky.

Časový rozvrh

- 09:00 – 10:30 – blok 1
- 10:30 – 10:45 – přestávka
- 10:45 – 12:00 – blok 2
- 12:00 – 13:00 – oběd
- 13:00 – 14:15 – blok 3
- 14:15 – 14:30 – přestávka
- 14:30 – 15:45 – blok 4
- 15:45 – 16:00 – přestávka
- 16:00 – 17:00 – blok 5

Agenda

- Úvodní informace
- Standardní metody řízení projektu
- Požadavky na produkt, plány, rizika
- Agilní metody řízení projektu – trend vývoje software
- Metoda Scrum
- Metoda XP
- CMMI – základní informace, vztah k projektovému řízení
- Diskuse

Standardní metody řízení projektů

- V současné době existují uznávané a hojně používané standardy v projektovém řízení
- Aktivity související s projektovým managementem se soustřeďují kolem dvou organizací, které vydávají publikace a poskytují certifikace
  - IPMA (International Project Management Association)
  - PMI (Project Management Institute)
- Jednou z nejkomplexnější a nejpoužívanější publikací popisující podrobně problematiku projektového managementu je „A guide to the Project Management Body of Knowledge“ vydaný PMI

Základní pojmy a definice

Co je projekt?

- Definicí pojmu projekt existuje celá řada
- PMBOK definuje projekt jako:
  - „A project is a temporary endeavor undertaken to create an unique product, service or result“
  - „Projekt je dočasně vynaložené úsilí k vytvoření unikátního produktu, služby nebo výsledku.“
- Podobnou definici má také norma ISO 10006 (Směrnice jakosti v managementu projektu)
  - „Projekt je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení předem stanoveného cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji.“

Základní pojmy a definice

- **Project manager (Projektový manager)**
  - Osoba zodpovědná za vedení projektu
- **Customer/user (Zákazník/uživatel)**
  - Osoba nebo organizace, která bude používat výsledky projektu
  - V některých případech pojmy „customer“ a „user“ jsou synonyma
  - Obecně je „customer“ osoba nebo organizace, která se stane vlastníkem výsledku projektu, zatímco „user“ je někdo, kdo přímo využívá výsledky projektu
- **Performing organization (Prováděcí organizace)**
  - Firma, podnik, organizace jejíž zaměstnanci vykonávají práce na projektu
- **Project team members (Členové projektového týmu)**
  - Je skupina lidí, která provádí práce na projektu

Základní pojmy a definice

- **Project management team (tým zabývající se projektovým managementem)**
  - **Členové projektového týmu, kteří vykonávají aktivity související s řízením projektu**
- **Project Sponsor (sponzor projektu)**
  - **Osoba nebo skupina osob poskytující zdroje pro financování projektu**
- **Stakeholder (podílník)**
  - **Osoba nebo organizace která je aktivně zapojena do projektu nebo osoba či organizace, jejíž zájmy mohou být ovlivněny výsledkem projektu – definice podle PMBOK**
  - **Osoba nebo organizace mimo projekt, která má zájem na výsledcích projektu – jiná definice**

Základní pojmy a definice

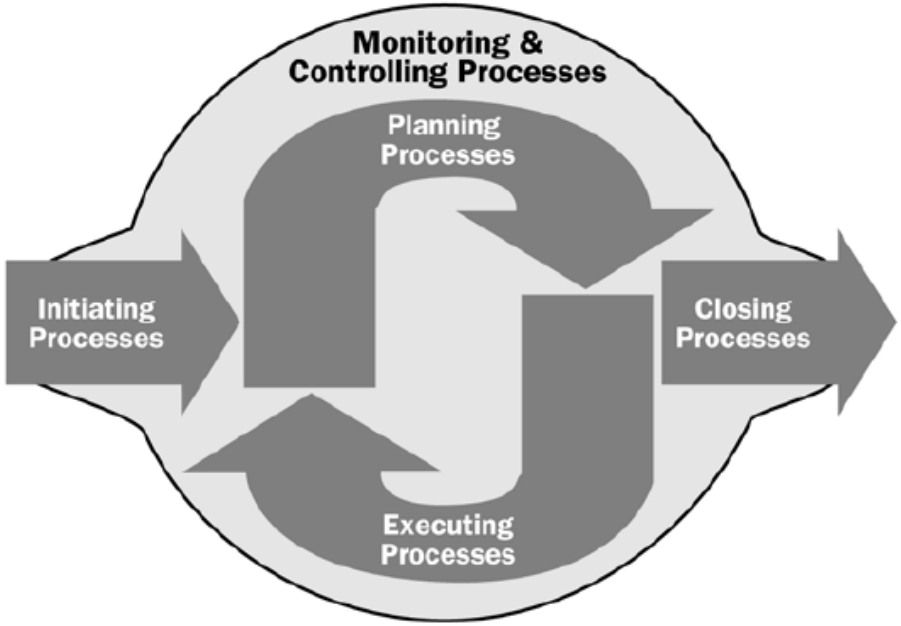
- **Životní cyklus projektu (Project Life Cycle)**
  - Soubor obecně po sobě následujících fází projektu, jejichž jména a počet je definován potřebami organizace nebo organizací zapojených do projektu.
  - Má začátek a konec
  - Obvykle se dělí do 4 fází
    - Iniciace
    - Plánování
    - Vykonávání
    - Ukončení

Základní pojmy a definice

- **Životní cyklus projektu (Project Life Cycle)**
  - Skupiny procesů jsou mapovány do jednotlivých fází životního cyklu
  - Procesy projektu obvykle můžeme rozdělit do 5 skupin
    - Iniciační/zahajovací procesy (Initiating Processes)
    - Plánovací procesy (Planning Processes)
    - Prováděcí procesy (Executing Processes)
    - Monitorovací a kontrolní procesy (Monitoring & Controlling Processes)
    - Ukončovací procesy (Closing Processes)

Základní pojmy a definice

- **Životní cyklus projektu (Project Life Cycle)**



Základní pojmy a definice

- **Iniciace a zahájení projektu**
  - Hlavním účelem této fáze projektu je vytvoření projektového záměru, základní definice projektu obsažené v zakládajícím dokumentu projektu (angl. Project Charter) a určení předběžného rozsahu projektu
  - Vytvoření projektového záměru slouží k získání autorizace pro jeho realizaci
- **Plánování projektu**
  - V této fází vycházíme ze základní listiny projektu vytvořené v předcházejícím kroku a provádíme detailní rozbor odhadu času, nákladů, lidských zdrojů, technologií a metodologií
  - Patří zde také plán kvality a identifikace a řízení rizik
  - Výstupem této fáze je podrobný a závazný plán projektu

Základní pojmy a definice

- **Provádění projektu**
  - Skupina prováděcích procesů je souhrnem všech aktivit, které jsou využívány k tomu, aby bylo dosaženo cílů projektů podle vytvořených plánů a ve stanoveném rozsahu
  - Jeho součástí je přímé řízení, podpora kvality, motivace členů týmu, distribuce informací a spolupráce s prodejci, případně zákazníkem
- **Monitorování a řízení**
  - Zaměřuje se na průběžné monitorování stavu projektu porovnáváním aktuálního stavu s plány a to z pohledu cílů projektu, času, nákladů a rizik
  - Také sem patří řízení změn, ověřování, že výstupy odpovídají požadavkům, spolupráce s okolím atd.
- **Uzavření projektu**
  - Vyvrcholení projektového snažení, které má své náležitosti, jako je kontrola podle dohodnutých smluv, akceptace dodaných výsledků zákazníkem, závěrečná fakturace, atd

Znalostní oblasti projektového managementu

- PMBOK dělí problematiku projektového managementu do 9 znalostních oblastí
- **Oblast Integration Managementu**
  - Obsahuje procesy a aktivity potřebné k identifikaci, definici, a koordinaci různých procesů a manažerských aktivit v rámci skupin procesů projektu
  - Jde například o vytváření projektu, prvního návrhu rozsahu, ale také o vytváření plánu a řízení projektu včetně kontroly integrace změn a uzavření projektu
- **Oblast Scope Managementu**
  - Obsahuje procesy potřebné k zajištění toho, že projekt bude obsahovat veškerou a pouze potřebnou práci k jeho úspěšnému dokončení
  - Project scope management je primárně zaměřen na definování a řízení toho, co v projektu obsaženo je a co není
  - Jsou v něm zahrnuty procesy jako plánování a definice rozsahu, seřazení práce a ověření a kontrola rozsahu

Znalostní oblasti projektového managementu

- **Oblast Time Managementu**
  - Procesy obsažené v project time managementu jsou potřeba k zajištění toho, aby byl projekt dokončen včas
  - Jde o definování, seřazení a odhad doby trvání aktivit, odhady čerpání zdrojů a kontrola časového plánu
- **Oblast Cost Managementu**
  - Oblast cost managementu obsahuje procesy zapojené do plánování, odhadování, rozpočtování a kontroly nákladů, aby projekt byl dokončen s daným rozpočtem
- **Oblast Human Resource Managementu**
  - Human resource management obsahuje procesy k organizaci a řízení projektového týmu. Tedy procesy jako: plánování lidských zdrojů, zajištění, vytvoření a řízení projektového týmu

Znalostní oblasti projektového managementu

- **Oblast Quality Managementu**
  - Procesy quality managementu obsahují všechny aktivity vykonávané organizací ke stanovení pravidel, cílů a odpovědností k tomu, aby projekt splnil požadavky pro které byl založen
  - Obsahuje procesy týkající se plánování kvality, kde se definují jednotlivé relevantní standardy a určuje se, jak jich dosáhnout
  - Dále pak obsahuje procesy týkající se zajišťování podpory kvality, kde se aplikují plánované aktivity k zajištění toho, aby projekt využil všechny procesy k dosažení cílů
  - Nedílnou součástí je rovněž provádění kontroly kvality, kdy se monitorují dané výstupy projektu, jestli splňují standardy kvality

Znalostní oblasti projektového managementu

- **Oblast Communications Managementu**
  - Pokrývá procesy potřebné k zajištění včasné a odpovídající tvorby, spojování, distribuci, ukládání, získávání a konečnou distribuci informací o projektu
  - Jde o procesy jako plánování komunikace (určení informací pro zainteresované osoby), distribuce informací (zajištění dostupnosti informací pro zainteresované osoby), reportování vývoje (stav projektu, rychlost postupu a předpovědi), atd

Znalostní oblasti projektového managementu

- **Oblast Risk Managementu**
  - Project risk management obsahuje procesy spojené s vedením plánu rizik, identifikací, analýzou, reakcemi a monitorováním a kontrolováním projektu. Jedná se o:
    - vytváření plánu řízení rizik
    - identifikaci rizik
    - kvalitativní analýzu rizik
    - kvantitativní analýza rizik
    - plánování reakcí na rizika
    - plánování předcházení rizikům
    - monitorování a kontrola rizik

Znalostní oblasti projektového managementu

- **Oblast Procurement Managementu**
  - Obsahuje procesy k zakoupení nebo získání produktů, služeb nebo výstupů potřebných pro práci projektového týmu zvenčí. Jedná se o:
    - Plán nákupů a akvizic
    - Plán kontraktů
    - Vyjednávání s prodejci
    - Výběr prodejce
    - Administrace smluv

Requirements management

- Úkolem „requirements managementu“ je
  - Sběr a správa požadavků na výsledný produkt nebo na části produktu
  - Odhalit rozpory mezi požadavky a plánem projektu a produktem
- Kroky získávání požadavků na produkt
  - Seznam specifikací od zákazníka nebo jeho zástupce
  - Fáze zkoumání a analyzování požadavků za účelem porozumění zákazníkových potřeb
  - Seznam požadavků je dobré detailně revidovat, ale žádný originální požadavek nemodifikovat nebo dokonce nevymazat ze seznamu
  - Komunikace se zákazníkem může probíhat v několika iteracích – upřesňování požadavků
  - Dokument obsahující finální verzi požadavků logicky členěnou do sekcí
  - Odeslání zákazníkovi k podpisu
  - Předání požadavků organizaci – revize a implementace

Requirements management

- Dobré praktiky při psaní požadavků
  - Definice požadavků se zaměřuje na otázku CO a né JAK
  - Ve fázi definice požadavků je podstatné jaká vlastnost se má realizovat a méně podstatné, jak se daná vlastnost bude realizovat
  - Požadavky jsou obvykle psané přirozeným jazykem, flexibilita jazyka a vyjadřování pisatele může vést k nedorozumění a následným problémům při realizaci
  - Typické problémy
    - Dokument s požadavky obsahuje vágní terminologii, která vede k nejasnostem
    - Požadavky jsou spíše naznačeny než jasně specifikovány
    - Dokument nemá žádnou strukturu – obtížná orientace
    - Několik požadavků je smícháno do jednoho záznamu
  - Né všechny požadavky musí být nutně splněny při realizaci – jé dobré je rozdělit na „Critical“ a „Desired“
    - „Critical“ – musí být realizovány
    - „Desired“ – by měly být realizovány, pokud je to možné

Requirements management

- Organizace požadavků v rámci dokumentu
  - Setřídění požadavků v rámci dokumentu do skupin (funkční, hardware, software, nástroje, dokumentace, atd)
  - Každý požadavek musí mít unikátní referenci (číslo, kombinace znaku a čísla, atd.)
  - Pouze jeden konkrétní požadavek by měl být obsažen v jedné větě – podpora trasovatelnosti požadavků
  - Vyhnout se komplikovaným větám, podmínkám a odvozeným požadavkům

Requirements management

Typ požadavku	Specifikace požadavku	Pož. ID
Business	Vytvovřit knihovnu funkcí - *.lib	C1
	Finální release – 30. června 2010	D1
Implementace	Kódování v jazyce C podle ANSI C standardu	C2
	Použitý kompilátor – Metrowerks CodeWarrior v.8.3	C3
	Kódování v assembleru	D2
Hardware	Knihovna musí běžet na DSC56F8006	C4
Funkce	Luenbergrův pozorovatel	C5
	Clarkova transformace	C6
	PID regulátor	C7

Plánování

- Detailní časový plán obsahuje klíčové informace pro sestavení harmonogramu projektu
- Je východiskem pro koordinaci všech projektových prací a vytváří podklad pro měření stavu plnění projektu
- **Osvědčené praktiky:**
  - základem je mít zpracované vstupní požadavky projektu
  - rozdělit komplexní úkoly na podúkoly, jejichž trvání lze snadněji ohodnotit
  - svolat brainstorming klíčových členů týmu případně externích odborníků
  - připravit podklady z podobných projektů, pokud takové existují

Plánování

- **Časový plán projektu by měl obsahovat:**
  - Důležité termíny projektu
  - Milníky
  - Přehledně zpracovaný sled navazujících úkolů
  - Údaje o předpokládané délce trvání jednotlivých úseků činnosti
  - Vazby a souslednosti mezi jednotlivými úseky práce
  - Návaznosti s ohledem na zdroje, které se předpokládají, že budou daný úkol vykonávat
  - Informace o zdrojích, které jsou požadovány pro splnění úkolu, jako jsou lidské zdroje, zařízení, přístroje atd.
  - Informace, kdy by měly být splněny ucelené úseky, které spolu úzce souvisí
  - Termíny, kdy bude projekt vnitřně auditován – aktuální stav bude srovnáván s plánem
  - Jiné informace napomáhající sledování a údržbě časového harmonogramu v návaznosti na procesy koordinace, monitorování a řízení po dobu životního cyklu projektu

Plánování - metody

- Časový rozpis projektu - obvykle představovaný diagramy a harmonogramy
- Diagramy a plánovací techniky prodělaly velký rozvoj v minulém století
- Nedostatky jednoduchých tzv Ganttových diagramů a diagramů milníků vyřešily síťové grafy
- Rozvojem výpočetní techniky a softwarové podpory dochází k modifikacím klasických metod jako jsou Ganttovy diagramy a vlastnosti, které omezovaly jejich použití jsou implementovány
- Metody založené na síťových grafech
  - PERT (Project Evaluation and Review Technique)
  - Metoda kritické cesty – CPM (Critical Path Method)
  - Metoda šípkových diagramů – ADM (Arrow Diagram Method)
  - Metoda síťových diagramů s rozšířenými možnostmi vazeb – PDM (Precedence Diagram Method)
  - Metoda grafického hodnocení a kontroly projektu – GERT (Graphical Evaluation and Review Technique)

Plánování – Ganttovy diagramy

- Díky jednoduchosti tvorby jsou často používané a pro jejich pochopení není potřeba zvláštní kvalifikace
- Diagramy mají však ve své původní podobě slabiny:
  - neukazují závislosti mezi úkoly
  - změna v délce nebo začátku jednoho úkolu se nepromítá do zbývající části diagramu

	T1	T2	T3	T4	T5	T6	T7
Úkol 1	x	x	x				
Úkol 2		x	x				
Úkol 3		x	x	x	x		
Úkol 4				x	x		
Úkol 5			x	x	x	x	
Úkol 6						x	x

Plánování – diagram milníků

- Milník (milestone) je časový údaj, který se váže k nějaké události
- Diagramy milníků jsou v podstatě zjednodušení Ganttova diagramu
- V praxi se spíše používají zobrazení ve formě tabulek

Milník	**Datum**
Zahájení projektu	3. 1. 2010
Příprava požadavků na produkt	31. 1. 2010
Plán projektu	14. 2. 2010
Příprava specifikace pro 1. fázi projektu	20. 2. 2010
Realizace 1. fáze	30. 3. 2010

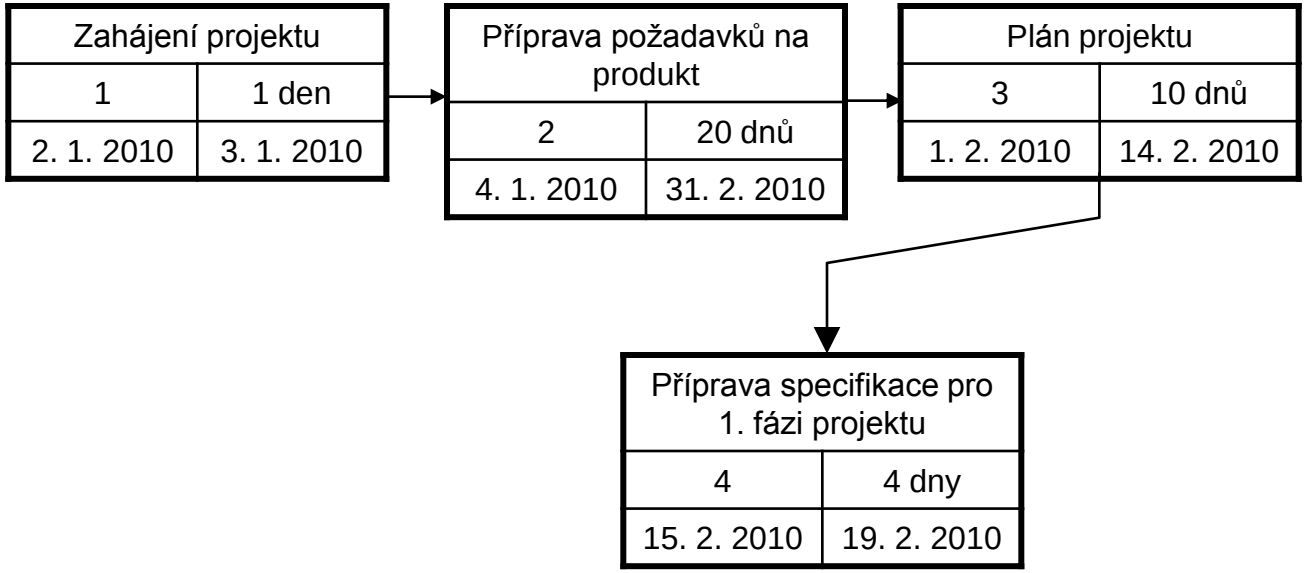
Plánování – PERT a CPM

- Obě metody využívají k zobrazení metod síťových grafů
- CP je metoda založená na vyhledávání a analýze kritické cesty projektu, což je nejdelší sled úkolů, které neobsahují žádné časové rezervy (kritická cesta je definována jako (časově) nejdelší možná cesta z počátečního bodu grafu do koncového bodu grafu)
- PERT je zobecněním metody kritické cesty (CPM)
  - Tato metoda se používá k řízení složitých akcí majících stochastickou povahu
  - Doba trvání každé dílčí činnosti chápe jako náhodná proměnná mající určité rozložení pravděpodobnosti
  - Empiricky bylo zjištěno, že v praxi toto nejlépe vystihuje tzv. *beta rozdělení* - nejlépe vystihuje proměnlivost provozních podmínek
- Klíčový rozdíl v analýze PERT, na rozdíl od jiných metod analýz je, že doba trvání každé činnosti je odvozena statisticky

Plánování – PERT a CPM

- Odhad činnosti se provádí s použitím tří časů:
  - nejoptimističtější odhad
  - nejpesimističtější odhad
  - nejpravděpodobnější odhad
- Vypočtená doba trvání činnosti je daná následujícím vztahem:
  - **Te_{50} = (a + 4m + b) / 6**
  - kde:
  - **a - optimistický odhad**
  - **b - pesimistický odhad**
  - **m - nejpravděpodobnější odhad**
  - **Te - odhad doby trvání**
  - **Te_{90} = Te_{50} + 1,38 * (b - a) / 6 = (4 * m + 2,38 * b – 0,38 * a) / 6**

Plánování – PERT a CPM



Risk management

- V okamžiku, kdy jsou detailně naplánovány všechny aktivity projektu a zdroje, je čas pro identifikování a hodnocení potenciálních rizik projektu
- Risk plán obsahuje seznam všech předvídatelných rizik a akce, které působí preventivně proti vzniku rizik, případně omezují jejich dopad
- Kompletní plán rizik obsahuje:
  - Seznam předvídatelných rizik projektu
  - Kvantitativní ohodnocení pravděpodobnosti, že příslušné riziko nastane
  - Popis případných dopadů na projekt pokud příslušné riziko nastane
  - Kvantitativní ohodnocení závažnosti jednotlivých rizik
  - Seznam preventivních akcí, které vedou ke snížení pravděpodobnosti, že příslušné riziko nastane
  - Seznam podmíněných akcí, které zmenšují dopad na projekt, pokud příslušné riziko nastane
  - Proces pro managování rizik po dobu životního cyklu projektu
- Každý risk musí mít unikátní identifikační číslo

Risk management – kvantifikace rizik - pravděpodobnost rizika (likelihood)

Title	Score	Description/Popis
Very low – velmi nízká	20	Vysoce nepravděpodobný vznik rizika na základě aktuálních informací, okolnosti způsobující příslušné riziko jsou rovněž nepravděpodobná
Low – nízká	40	Nepravděpodobný vznik rizika. Nicméně je nutno riziko monitorovat, jelikož za jistých okolností se stává více pravděpodobné, že příslušné riziko může nastat
Medium – střední	60	Pravděpodobné, že riziko nastane
High – vysoká	80	Velmi pravděpodobné, že riziko nastane, na základě okolností projektu
Very high – velmi vysoká	100	Vysoce pravděpodobné, že riziko nastane

Risk management – kvantifikace rizik - dopad / vliv rizika (impact)

Title	Score	Description/Popis
Very low – velmi nízká	20	Bezvýznamný/nepatrný vliv na projekt. Dopad na projekt je nemožné změřit, protože je natolik bezvýznamný
Low – nízká	40	Nepatrný vliv na projekt. Má za následek méně než 5% odchylku na rámec projektu, plánovaný konec projektu nebo rozpočet
Medium – střední	60	Měřitelný vliv na projekt. Má za následek 5% - 10% odchylku na rámec projektu, plánovaný konec projektu nebo rozpočet
High – vysoká	80	Významný vliv na projekt. Má za následek 10% - 25% odchylku na rámec projektu, plánovaný konec projektu nebo rozpočet
Very high – velmi vysoká	100	Závažný vliv na projekt. Má za následek více než 25% odchylku na rámec projektu, plánovaný konec projektu nebo rozpočet

Risk management

- Seznam risků obvykle ve formě tabulky obsahuje
  - Kategorii risků – zpřehledňuje seznam
  - Popis jednotlivých risků
  - Unikátní identifikační číslo
- Priority
  - Skóre (bodové hodnocení) definující prioritu může být vypočteno následovně

priorita = (pravděpodobnost + dopad) / 2

- Každý risk je pak ohodnocen a jsou definovány priority
- Vytvoření plánu risků
  - Ve formě přehledné tabulky

Risk management – tabulka risků

Kategorie risku	Popis risku	Risk ID
Požadavky	Nejasná specifikace požadavků na produkt	1.1
	Dosažení maximálního momentu je limitováno hardware	1.2
Plán	Plán neobsahuje všechny úkoly a aktivity	2.1
	Plán neobsahuje přesné závislosti	2.2
Zdroje	Lidé přiřazení na projekt nemají dostatek znalostí	3.1
	Nedostatečné technické vybavení pro práci na projektu	3.2
	Materiál použitý na projekt je těžko dostupný	3.3
Rozpočet	Rozpočet bude překročen	4.1

Risk management – hodnocení priorit

Priorita - body	Priorita - rating	Priorita - barva
0 – 20	Velmi nízká	Bílá
21 - 40	Nízká	Zelená
41 - 60	Střední	Žlutá
61 - 80	Vysoká	Oranžová
81 - 100	Velmi vysoká	Červená

Risk management

Risk ID	Pravděpodobnost	Dopad	Priorita	Rating
1.1	20	80	50	Střední
1.2	20	100	60	Střední
2.1	20	60	40	Nízký
2.2	20	20	20	Velmi Nízký
3.1	80	80	80	Vysoký
3.2	20	40	30	Nízký
3.3	60	80	70	Vysoký
4.1	20	80	50	Střední

Risk management - risk plán

Risk ID	Rating	Následky/ ovlivněná oblast	Preventivní akce	Podmíněné akce	Vyvolání	Zodpovědná osoba
1.1	Střední	Funkčnost, plán	Důkladná revize zákaznické specifikace	Upřesnění od zákazníka	Neporozumění specifikaci	J.L.
1.2	Střední	Změna specifikace HW, plán, funkčnost	Zjištění skutečného stavu před započetím realizace	Řešit problém se zákazníkem – změna HW nebo změna specifikace	Testování – nelze docílit max. moment	J.L.
2.1	Nízký	Plán	Důkladná revize plánu	Změna plánu	Zákaznická specifikace vyžaduje činnosti, které nejsou naplánovány	J.L.

Testování software

- Během vývoje vývojář provádí unit test periodicky při každé úpravě kódů
- Důležitým bodem je, že vývojář v průběhu debugování nachází chyby, kód průběžně opravuje a ukládá test vektory, kde byla nalezena chyba
- Po ukončení vývoje kódu, přechází zdrojový kód do fáze testování
- Paralelně s vývojem kódu je vytvářen testovací plán a testovací procedura
  - Jedná se o dokument, který se pak bere jako vodítko pro testera
  - Definuje způsob testování, důkladnost a použité metody

Testování software

- Generují se testovací vektory, případně „testovací trajektorie“
- Ty by měly být sestaveny takovým způsobem, aby reflektovaly
  - Boundary value analysis
  - Range issue analysis
  - Discontinuity points analysis
  - Control flow analysis
  - Data flow anlysis

Testování software

- **Prováděné testy**
  - **Funkční testy – deterministické (white box testing)**
    - **Testovácí vektory jsou navrženy po důkladné analýze**
    - **Mají zahrnovat kritické oblasti**
    - **Hraniční podmínky**
    - **Nespojitosti**
    - **atd.**
  - **Funkční testy – stochastické (black box testing)**
    - **Testovací vektory vygenerované náhodně**
  - **Regresní testy**
    - **Test vektory, které napomohly odhalit chyby při vývoji**
  - **Fyzikální testy**

Testování software

- **Prováděné testy**
  - **Testy výkonu (performance tests)**
    - **Ve smysly počtu cyklů procesoru**
    - **Ve smyslu obsazené paměti programu a dat**
  - **Test coverage**
    - **Statement coverage (každý řádek kódu) – 100%**
    - **Condition coverage (každá podmínka v kódu) -100%**
    - **Path coverage (všechny možné cesty)**
  - **Test coverage by měla být měřen automatickým nástrojem (v assembleru problém)**

Testování software

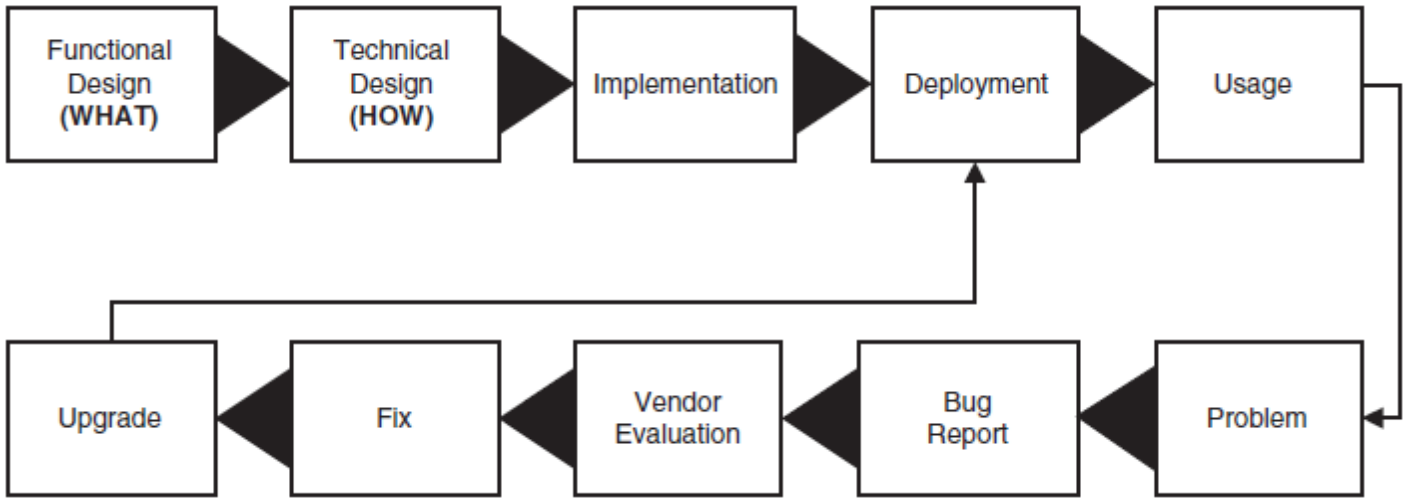
- Testy systémové
- Testy akceptační
- Obvyklé dokumenty
  - Testovací plán a procedury
  - Revizi testovacího plánu a procedury
  - Testovací report

Životní cyklus – modely – Build-and-Fix

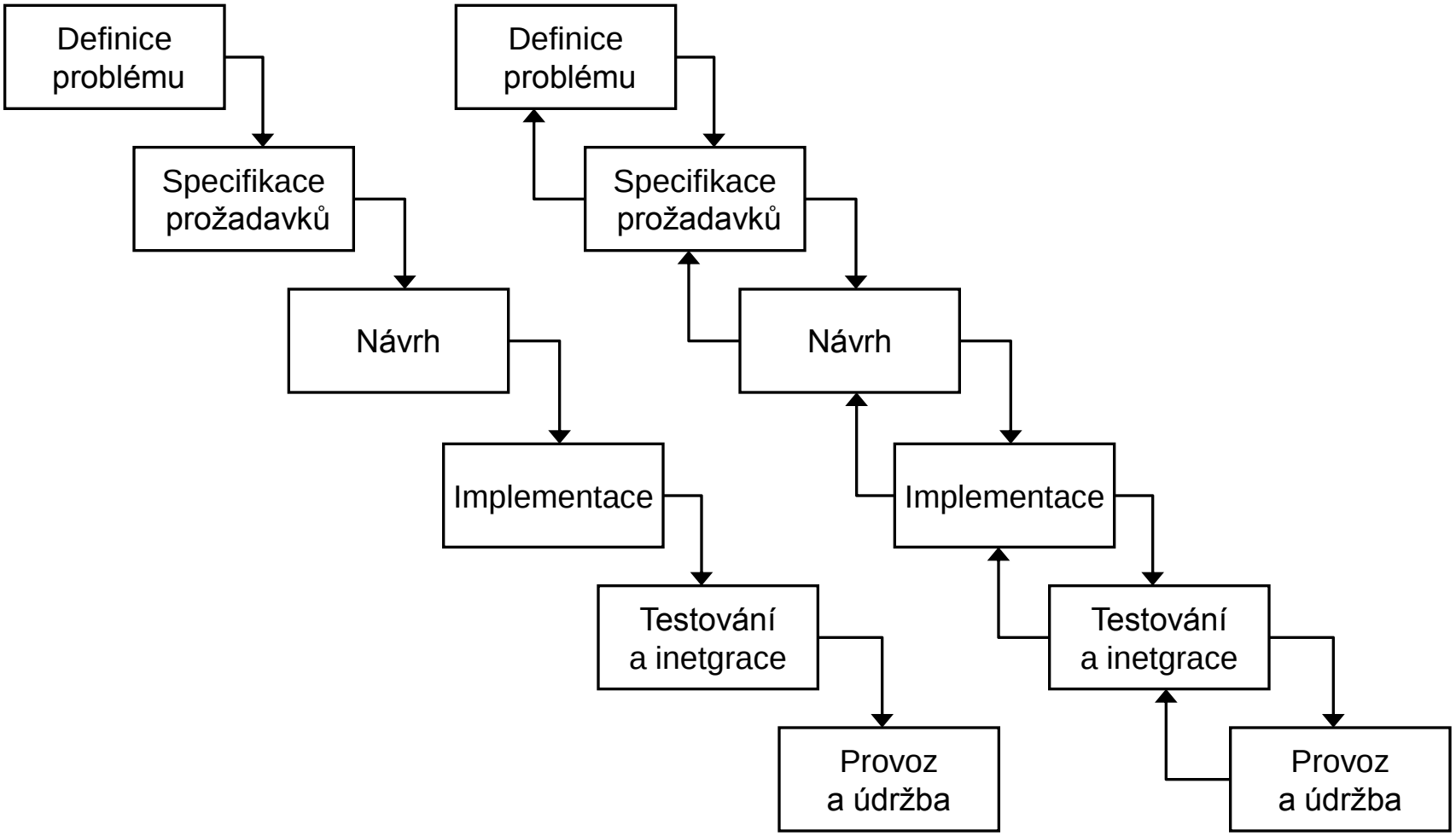
- Práce bez metodologie = Build-and-Fix model
- Tato metoda jen zřídka vyprodukuje užitečný výsledek
- Její použití je nebezpečné
- Prakticky nemožné hodnotit
  - Pokrok v projektu
  - Kvalitu
  - Risky
- Výhody
  - Pouze kódování
  - Žádné plánování, dokumentace, standardy
  - Vyžaduje pouze minimální zkušenosti

Životní cyklus – modely – Build-and-Fix

- **Nevýhody**
  - **Nebezpečné**
  - **Žádná kvalita, žádné risky**
  - **Často musí být výsledky práce vyhozeny**
  - **Neustále update software – obtěžování zákazníka**



Životní cyklus – modely – Waterfall



Životní cyklus – modely – Waterfall

- Klasický vývojový model
- Skládá se z jednotlivých fází, bez vzájemné interakcí
- Iterace je považována za ukončenou až po zpracování dokumentace pro danou fázi
- V extrému se považuje dokumentace za nástroj komunikace mezi členy týmu a zpracovatelem a zákazníkem
- Na konci každé fáze – schvalovací řízení

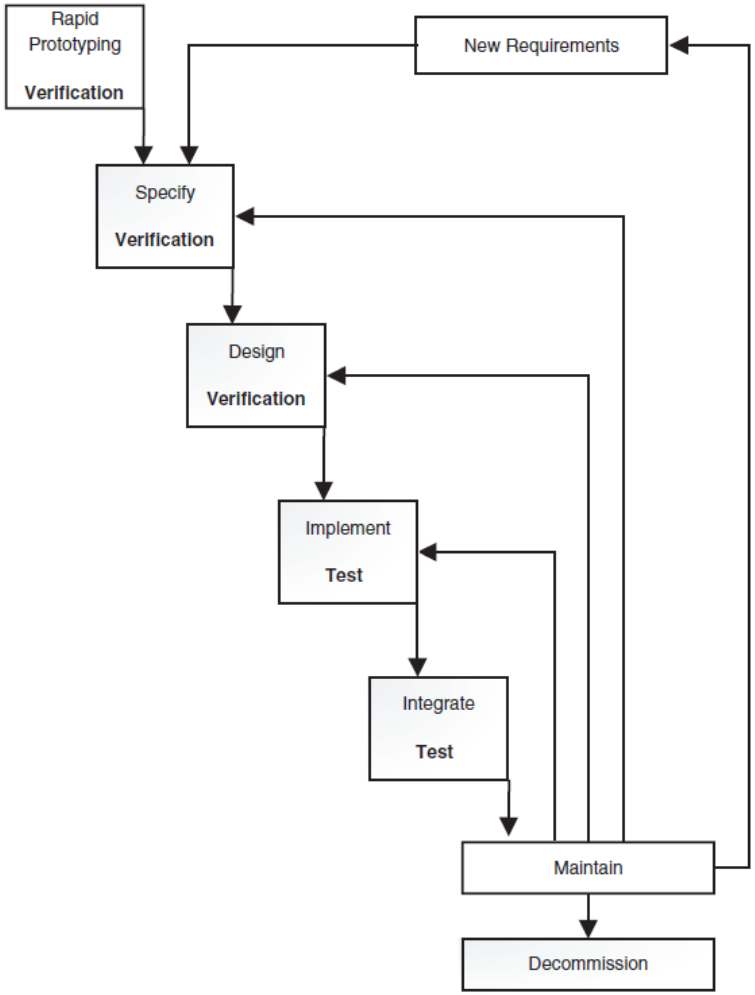
Životní cyklus – modely – Waterfall

- Výhody
  - Jednoduchý model – definuje vývoj krok za krokem
  - Funguje dobře pro technicky slabé nebo nezkušené zaměstnance
  - Dá se napasovat na různé typy projektů
  - Projekt se jednoduše řídí – víme, ve které fází se nachází
  - Produkuje podrobnou dokumentaci
- Nevýhody
  - Žádná flexibilita
  - Je obtížné nalezenou chybu řešit během životního cyklu
  - Nevhodné pro vývoj – nereflektuje změny během procesu vývoje
  - U komplexnějších projektů jednoduchost není vhodná
  - Model nepočítá s modifikací požadavků za běhu projektu

Životní cyklus – modely – Rapid prototyping

- Je používán pro jednorázový vývoj software
- Velmi podobný Waterfall modelu
- Účelem modelu je rychle vytvořit prototyp za účelem
  - Otestovat implementační metodu
  - Jazyk
  - Přijatelnost řešení zákazníkem
- V případě, že daná technologie je použitelná, slouží prototyp jako základ vývoje produktu

Životní cyklus – modely – Rapid prototyping



Životní cyklus – modely – Spiral model

- Spirálový model byl vyvinul pan Dr. Barry Boehm z TRW
- Je to v podstatě rozšířením Waterfall/Rapid prototypového modelu přidáním risk analýzy před každou fázi kaskády
- Lze si Spiral model představit jako Rapid prototyping model nakreslený ve formou spirály
- Úspěšně otestovaný na velkých projektech, speciálně v případech znovu použití již vytvořených celků
- Je velmi dobře popsaný panem Boehmem

Životní cyklus – modely – Spiral model

Agilní metody – důvody vzniku

- **Tlak současné ekonomiky na zkracování životního cyklu**
- **Co nejrychlejší dodání a uvedení do provozu**
  - rychlost znamená konkurenční výhodu
- **Změna požadavků na software/systém v průběhu vývoje**
  - změny mohou být časté a mnohdy i zásadní
- **Zákazník nemá jasnou představu o produktu na počátku projektu**
  - software/systém je natolik komplexní, že podrobnou a přesnou specifikaci produktu je velmi obtížné vytvořit
- **Nicméně zákazník požaduje kvalitu řešení**
  - finální verze produktu musí splňovat poslední verzi několikrát změněných požadavků

Agilní metody – historie vzniku

- Snahy o nalezení nových metod řízení projektů, které by lépe odpovídaly novým požadavkům doby - zhruba od druhé poloviny 90 let minulého století
- V roce 2001 se začala formovat skupina předních odborníků v oblasti softwarového inženýrství, která se pokoušela řešit problémy vznikající při tvorbě software.
  - Sešli se v americkém Utahu s cílem analyzovat jednotlivé metodiky, na kterých pracovali a najít společné rysy a formulovat základní principy
  - Výsledkem setkání bylo sepsání dokumentu „Manifesto for Agile Software Development“.
  - Tento dokument se stal základem rodiny agilních metodik a přístupů.
  - V manifestu odborníci tvrdí, že našli lepší způsob vývoje software, sami jej používají a chtějí pomoci i ostatním, aby jej používali a říkají, že dávají přednost:

individualitám a komunikací před **procesy a nástroji**

fungujícímu software před **obsažnou dokumentací**

spolupráci se zákazníkem před **sjednáváním kontraktu**

reakci na změnu před **plněním plánu**

Agilní metody – základní principy

- „The Agile Manifesto“ definuje 12 základních principů

- Nejvyšší priorita je uspokojit zákazníka pomocí brzkých a průběžných dodávek software
- Jsou vítány změny požadavků i v pozdějších fázích vývoje - využívají změnu přinášející zákazníkovi konkurenční výhodu.
- Časté dodávání fungujícího software v rozsahu od několika týdnů do několika měsíců - preferované jsou kratší časové úseky.
- Obchodníci a vývojáři musí pracovat po celou dobu projektu společně každý den
- Vytvářet projekty s okruhem motivovaných jednotlivců. Zabezpečit pro ně prostředí a podporu, kterou potřebují a věřit jim, že danou práci zvládnou.
- Nejúčinnější a nejefektivnější metodou předávání informací v rámci vývojového týmu je osobní (face-to-face) komunikace.
- Fungující software je hlavním měřítkem pokroku projektu
- Agilní procesy podněcují udržitelné tempo vývoje. Sponzoři, vývojáři a uživatelé by měli být schopni toto tempo udržet po neomezenou dobu
- Neustálá pozornost směřovaná na technickou dokonalost a dobrý návrh zvyšuje agilitu
- Jednoduchost – umění maximalizovat množství práce, která se nemusí udělat je zásadní.
- Samoorganizující se týmy vynikají nejlepšími návrhy, architekturou i požadavky
- V pravidelných intervalech by tým měl uvažovat nad tím, jak může být více efektivní a přizpůsobit tomu chování.

Agilní metody

- Mezi agilní metodiky řadíme:
  - Extrémní programování (Extreme Programming, XP)
  - Scrum
  - Lean Development
  - Dynamic Systems Development Method (DSDM)
  - Feature–Driven Development (FDD)
  - Adaptive Software Development (ASD)
  - Crystal metodiky
  - Agilní modelování (Agile Modeling)

Agilní metody – vlastnosti a omezení

- Nejsou rozpracovány do velkých podrobností
- Jsou iterativního charakteru
- Speciální důraz na komunikaci se zákazníkem a uvnitř týmu
- Dokumentace není prioritou

- Podmínky použití v praxi
  - Aktivní podíl zákazníka na vývoji
    - Přítomnost v týmu po celou dobu životního cyklu projektu
  - Velikost týmu & projektu
    - Spíše malé projekty
    - 8 až 20 vývojářů (100 a více)
  - Kvalita vývojářů
    - znalosti, zkušenosti a morální hodnoty

- Omezení použití metodik pro:
  - distribuované prostředí vývoje
  - podporu subdodavatelů
  - vytváření znovupoužitelných artefaktů
  - vývoj ve velkém týmu
  - vývoj kritických aplikací
  - vývoj velkého komplexního softwaru

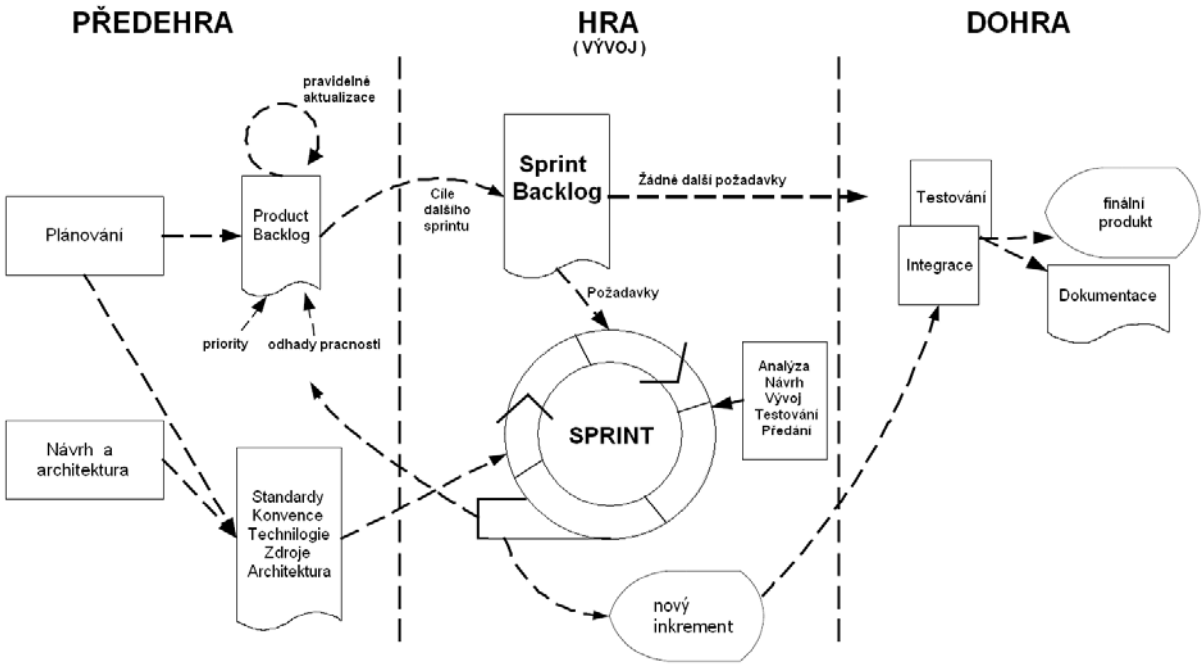
Scrum metoda - historie vzniku

- Metodika SCRUM Development Process (dále jen Scrum) byla představena Kenem Schwaberem a Mikem Beedlem v roce 2002
- Scrum metoda patří k nejmladším metodikám a přináší zcela nové praktiky do vývojového procesu
- Název „Scrum" není zkratka jak by se na první pohled mohlo zdát
  - pánové Schwabere & Beedle se nechali inspirovat hrou rugby, ke které přirovnávají vývoj software
  - termín „Scrum", v češtině překládáno jako „mlýn", právě z rugby pochází a znamená skrumáž lidí, kdy se hráči snaží dostat míč za čáru, tedy úspěšně dokončit projekt.
- Metodika vychází z předpokladu, že vývoj software je stejně tak jako rugby plný nečekaných událostí, změn a zvratů, na které musí tým pružně a rychle reagovat.

Scrum metoda – charakteristika

- *Scrum* byl vyvinut pro podporu řízení vývojového procesu
- Scrum
  - Není návod, jak programovat („programátorskou kuchřku“)
  - Nezabývá se konkrétními technologiemi, postupy či nástroji
- Zabývá se tím
  - Jak vést projekt během životního cyklu
  - Jak by měl tým při práci komunikovat a spolupracovat
  - Jak rychle a flexibilně reagovat na změny a jak co nejrychleji a nejefektivněji dosáhnout cíle projektu
- *Scrum*
  - Je tedy převážně manažerská metodika
  - Snaží vylepšit existující softwarově inženýrské praktiky a zaměřuje se na velice časté sledování a řešení všech překážek, které by mohly stát v cestě úspěšnému vývoji aplikace.
- Obecné vlastnosti metodiky *Scrum* se v podstatě shodují s ostatními agilními metodikami, které také vycházejí z Agilního manifestu

Scrum metoda – fáze vývoje



Scrum metoda – základní pojmy

- **Sprint**
  - Sprint je první typ iterace, který se pravidelně opakuje – obvykle 3 až 8 iterací
  - Délka trvání jednoho sprintu je přibližně jeden měsíc (zkušenosti ukazují, že sprint trvající 30 dní je dobrý kompromis mezi komfortním pracovním tempem a adaptabilitou)
  - Počet sprintů je různý podle typu projektu
  - Na začátku každého sprintu vybere Scrum Master spolu s týmem záznamy z *Product Backlogu*, které se budou implementovat v plánovaném *sprintu*
    - Výběr obvykle probíhá podle priorit definovaných zákazníkem a také podle funkčních a logických celků, na základě kterých je *Product Backlog* sestaven
    - Vybrané záznamy se pak přesunou do *Sprint Backlogu*. Tým pak odhadne pracnost a naplánuje průběh prací. Toto všechno se děje na plánovací schůzce sprintu (*Sprint Planning Meeting*). Plánovací schůzka by neměla přesáhnout osm hodin.
  - Na konci sprintu je opět schůzka celého týmu, kde se probere, co všechno se na projektu za tento sprint událo, jaké požadavky se povedlo splnit, jaké ne, a na jaké nové požadavky k zapracování se během vývoje přišlo.
- **Daily Scrum**
  - Jde o druhý typ iterace, který je o mnoho kratší - jeden den
  - Na začátku každého pracovního dne je uspořádána schůzka, tzv. *Scrum Meeting*
  - Díky takto krátkým iteracím je tým nejen neustále dobře informován, v jakém stádiu jsou jednotlivé práce na projektu, ale také je schopen případné problémy operativně řešit, protože na ně přijde brzy

Scrum metoda – základní pojmy

- **Backlog**
  - V rámci metodiky *Scrum* existují dva druhy těchto *Backlogů* - *Product Backlog* a *Sprint Backlog*
  - Product Backlog
    - Nejdůležitější dokumentem celé metodiky *Scrum*.
    - Tady jsou zaznamenány všechny požadavky zákazníka na software
    - Zákazník rovněž určuje prioritu pro každý požadavek
    - Položky *Product Backlogu* jsou průběžně aktualizovány, přidávány a odstraňovány, aby vždy odpovídaly aktuálnímu stavu požadavků
  - Sprint Backlog
    - Obsahuje položky z *Product Backlogu*
    - Co se má přesunout – volí tým ve spolupráci se *Scrum Masterem* na základě
      - Priorit specifikovaných zákazníkem
      - Logiky vývoje projektu
      - Položky *Sprint Backlog* jsou pak implementovány v jednom sprintu

Agilní metody – role a odpovědnosti

- **Metodika Scrum rozlišuje 6 rolí**

- **Scrum tým** (Scrum Team)
  - Je hlavní vývojová síla
  - Jeho zodpovědnost není pouze v implementaci
  - Měl by fungovat samorganizujícím způsobem
  - Hlavním úkolem je splnit cíle daného sprintu
  - Dalšími jeho úkoly je výběr konkrétních položek *Product Backlogu* do aktuálního sprintu a odhad pracnosti pro sestavení časového plánu
- **Zákazník** (Customer)
  - Podílí na sestavování seznamu položek pro *Product Backlog*
- **Management**
  - Stará se o obchodní jednání a uzavírání smluv
  - Svoji roli plní také v případě důležitých rozhodnutí ovlivňujících celý proces
  - Podílí se na sestavování požadavků a cílů celého projektu i jednotlivých etap

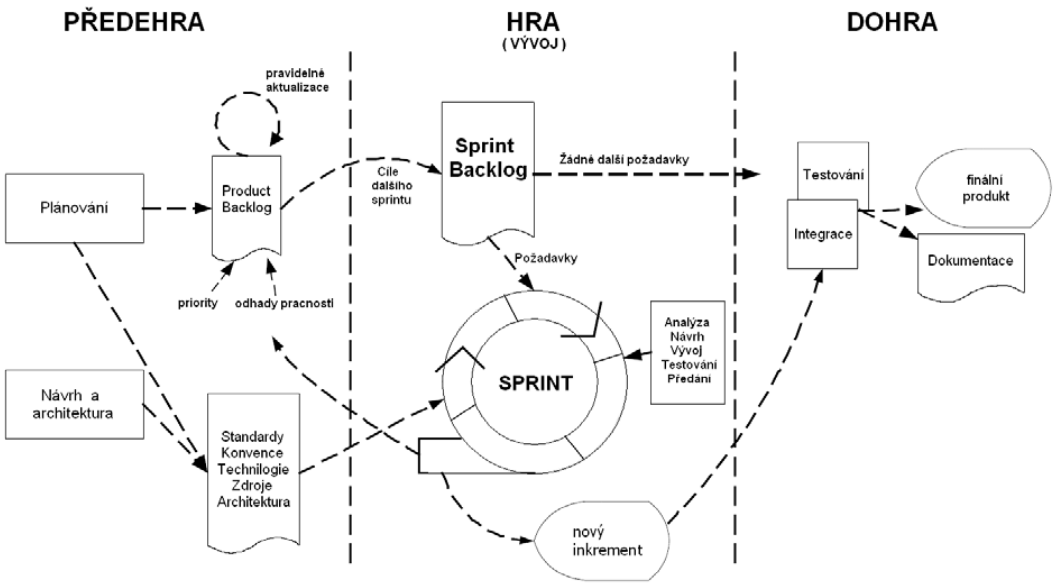
Agilní metody – role a odpovědnosti

- **Scrum Master**
  - Zodpovědnost za to, že proces vývoje probíhá v souladu s pravidly a praktikami metodiky *Scrum*.
  - Komunikuje vývojáři, se zákazníkem a s managementem
  - Jeho úkolem je zaručit, že všechny překážky zabraňující plynulému a produktivnímu vývoji budou co nejdříve odstraněny.
  - Pokud je *Scrum Master* nedokáže – musí najít schopného člověka

- **Vlastník produktu** (Product Owner)
  - Stojí na rozhraní mezi zákazníkem a dodavatelem
  - Je určován *Scrum Masterem*, zákazníkem a managementem
  - Jeho hlavní odpovědností je *Product Backlog*, odpovídá za úpravu a jeho publikaci
  - Má hlavní slovo v rozhodování o jeho položkách
  - Je zodpovědný za aktuální stav *Product Backlogu* a jeho dostupnost pro všechny členy týmu
  - Vlastník produktu se zavazuje nepřidávat další požadavky do *Sprint Backlogu* v průběhu sprintu
  - Položky *Product Backlogu* se často mění nebo přidávají, ale pouze mimo sprinty.

Scrum metoda – fáze vývoje

- Scrum metodika dělí proces tvorby produktu do tří hlavních fází - **předehra**, **hra** a **dohra**
- **Předehra** a **dohra** jsou fáze lineární
- **Hra** (prostřední fáze) se skládá ze sprintů, a ty se iterativně opakují.
- Scrum v průběhu celého vývoje nařizuje spoustu postupů, ale neuvádí jejich definici
- Říká například, kdy a kdo má naplánovat aktivity pro další sprint, ale jakým způsobem se má toto plánování provádět, to už součástí metodiky není a daný člen týmu už si musí konkrétní prostředky zvolit sám.



Scrum metoda – fáze vývoje - předehra

- Předehra se dělí do dvou podfází: **Plánování** a **Návrh a architektura**.
- **Plánování**
  - Obsahuje definici systému, který má být vytvořen
  - Je sestaven *Product Backlog* obsahující všechny známé požadavky
    - Požadavky mohou přicházet od zákazníka, obchodního nebo marketingového oddělení, zákaznické podpory nebo přímo od vývojářů
    - Požadavky jsou uspořádány podle priorit a je odhadnuto úsilí potřebné k jejich implementaci. Tyto odhady jsou v počátcích samozřejmě nepřesné, všechny hodnoty se v průběhu vývoje upravují a zpřesňují podle aktuální situace.
  - Sestavení projektového týmu, definovat vývojové nástroje, analýzu rizik, zdroje a postup schválení výsledného produktu.
- **Návrh a architektura**
  - Vytvoření návrhu systému na vysokém stupni abstrakce
  - Jako základ se bere *Product Backlog* vytvořený v předchozí fázi
  - Provádí se analýza a předběžné plány pro obsah jednotlivých verzí systému

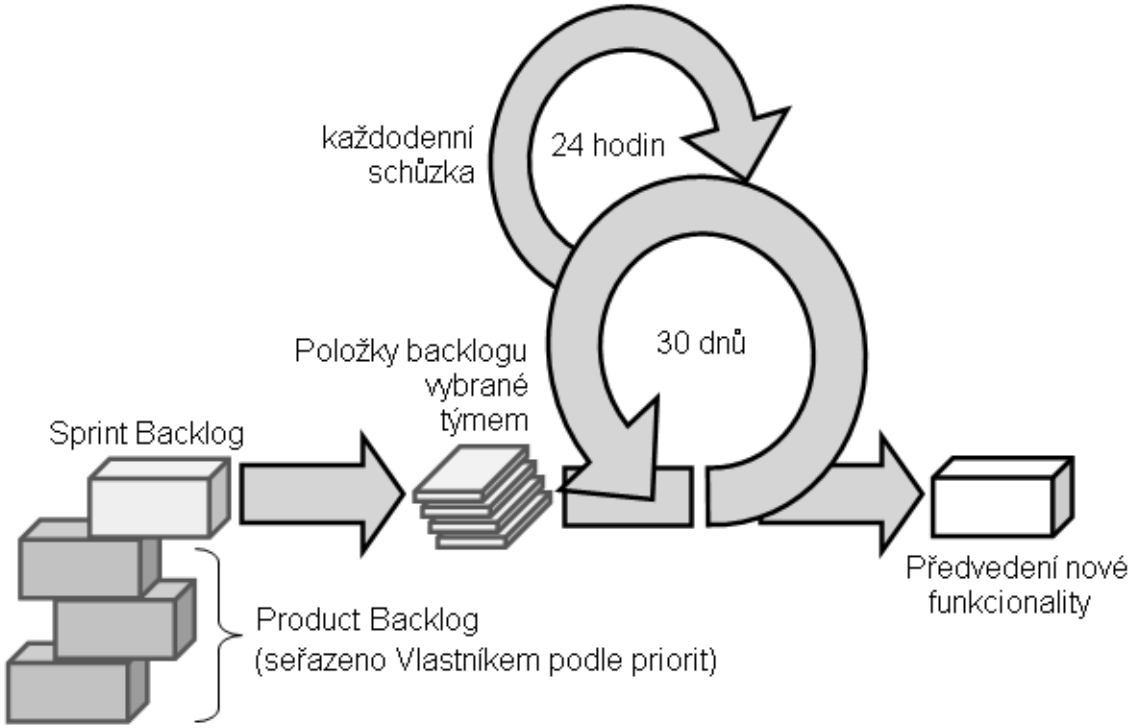
Scrum metoda – fáze vývoje - hra

- **Hra je hlavní částí, na kterou se *Scrum* zaměřuje**
  - **V této fázi je vytvářen produkt**
  - **Je hlavním zdrojem agility celého procesu *Scrum*, přímo se říká:**

> **Unpredictable is expected – Nepředvidatelné je očekáváno**

  - **Je členěna do tzv. *Sprintů* - tři až osm**
  - **Sprint trvá obvykle jeden měsíc**
    - **V rámci jednoho sprintu probíhají všechny klasické vývojové fáze: sbírání požadavků, analýza, návrh, vývoj, testování, předání**
    - **Každý sprint zpracovává část požadavků a funkčnosti zaznamenané v *Product Backlogu***
    - **Výsledek z každého sprintu je prezentován zákazníkovi**

Scrum metoda – fáze vývoje - hra



Scrum metoda – fáze vývoje - dohra

- Dohra
  - Fáze dohry nastává v okamžiku, kdy je kompletně zpracovaný celý *Product Backlog*
  - Důležité je, že zákazník již žádné požadavky nezadává a stávající požadavky nemodifikuje
  - Vytvoření finálního release produktu
  - Provádí se integrace celého systému
  - Probíhají závěrečné akceptační testy
  - Zajímavostí je, že dokumentace je vytvářena až v této fázi, v podstatě až na konci celého projektu

Scrum metoda – praktiky

- **Každodenní schůzka (Daily Scrum Meeting)**
  - **Nejtypičtější aktivita pro metodiku *Scrum***
  - **Na počátku každého dne se koná schůzka a ta má svá zvláštní pravidla**
  - **Svolává ji *Scrum Master* a koná se každý den ve stejnou dobu na stejném místě a trvá přibližně 15 až 30 minut.**
  - **Je vyžadována účast celého týmu. Kdo přijde pozdě, je pokutován *Scrum Masterem* a pokud přijde pozdě *Scrum Master*, je pokutován každým členem týmu**
  - **Schůzky se aktivně účastní *Scrum* tým, pasivně pak mohou být přítomni členové managementu, zástupci zákazníka, apod. Ti však mohou jen poslouchat a nesmějí se aktivně podílet na diskusi**
  - **Každý člen *Scrum* týmu musí odpovědět na tři otázky:**
    - **1. Co jsi udělal od posledního *Scrumu* (schůzky)?**
    - **2. Co budeš dělat do dalšího *Scrumu*?**
    - **3. Narazil jsi na nějaké překážky?**
  - **Při pokládání a odpovídání na tyto otázky se nesmí odbočovat a vždy by měla platit zásada, že mluví pouze jeden**
  - **Nalezené problémy se neřeší přímo na schůzce, pouze se zaznamenají a jejich řešení má později po skončení schůzky na starost skupina lidí, kterých se problém týká**

Scrum metoda – praktiky

- **Flexibilní předměty dodání (Flexible Deliverables)**
  - **Obsah dodávek není předem přesně dán**
  - **Závisí na typu projektu a prostředí, ve kterém nebo pro které se vyvíjí.**
- **Odhad pracnosti (Effort Estimation)**
  - **Pracnost jednotlivých položek *Product Backlogu* je odhadována průběžně**
  - **Podle informací z průběhu vývoje se hodnoty neustále zpřesňují**
  - **Určování odhadů má na starost vlastník produktu společně se členy týmu.**

Scrum metoda – praktiky

- **Plánovací schůzka sprintu (Sprint Planning Meeting)**
  - Je svolávána na začátku každého sprintu
  - Organizuje ji *Scrum Master* a má dvě fáze
  - První fáze se účastní zákazník, uživatelé, management, vlastník produktu a *Scrum tým*
    - Jejich úkolem je stanovit cíle nadcházejícího sprintu a funkcionality, které budou implementovány
    - Položky vybírají z *Product Backlogu* a sestaví z nich *Sprint Backlog*.
    - Sprint Backlog může obsahovat položky Product Backlogu detailněji rozpracované do více položek
  - Druhé fáze se účastní jen *Scrum Master* a *Scrum tým* a zaměřují se na konkrétní implementaci daných položek

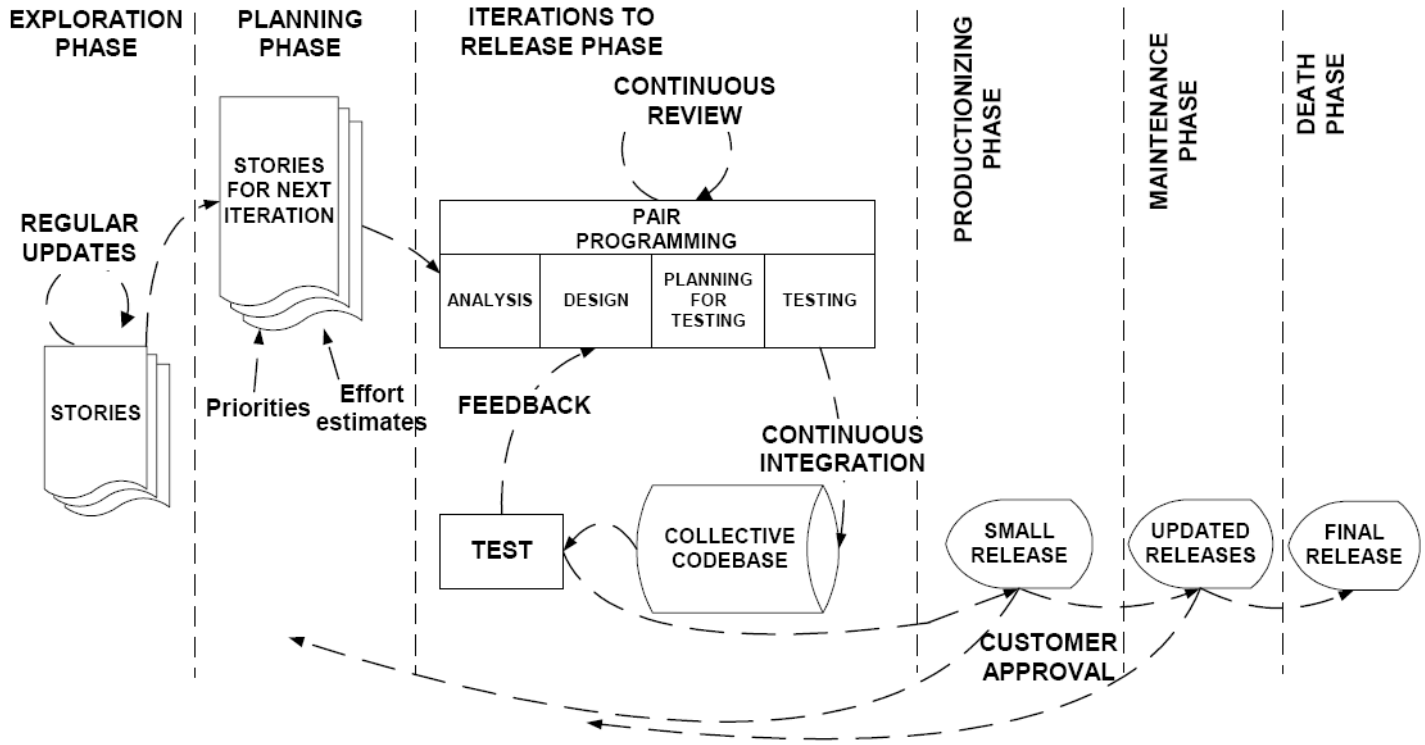
Scrum metoda – praktiky

- **Hodnotící schůzka sprintu (Sprint Review Meeting)**
  - **Svolává ji *Scrum Master* společně se *Scrum týmem* poslední den sprintu**
  - **Účastníci - vlastník produktu, management, zákazník a zástupci uživatelů**
  - **Předání výsledků sprintu zákazníkovi**
  - **Prezentuje se pouze to, co se skutečně udělalo, nesmí se ukazovat vlastnosti, které nejsou ještě hotovy**
  - **Schůzka by neměla být dlouhá a měla by být ohraničena čtyřmi hodinami**
  - **Rovněž příprava členů týmu by neměla přesáhnout čtyři hodiny**
  - **Schůzka by se měla řídit přísnými pravidly**
    - **Prezentovat začíná dodavatel a podává informace o tom, co bylo cílem sprintu, co se udělalo a co se může vyškrtnout z *Product Backlogu***
    - **Teprve pak dostava slovo zákazník a další zainteresované osoby**
    - **Následně probíhá diskuse**
    - **Zákazníkovy vstupy jsou zaneseny do *Product Backlogu***
    - **Na závěr schůzky Scrum Master oznámí příští termín.**

Extrémní programování (XP)

- Extrémní programování – (Extreme Programming - XP) je dnes asi nejznámějším zastupitelem agilních metodik - autorem je Kent Beck
- V době svého vzniku, v roce 1999, se snažilo reagovat na problémy tehdejších projektů a řešit je cestou extrémů
- Vychází z principu, který říká, že pokud něco funguje, tak to budeme používat v maximální možné míře (extrémně)
- Základním pravidlem, které ctí extrémní programování je:
  - „Jediným exaktním, jednoznačným, změřitelným, ověřitelným a nezpochybnitelným zdrojem informací je zdrojový kód“
- Extrémní programování se zaměřuje především na psaní zdrojového kódu
- XP je poměrně propracovaná metodika
- Jádro XP tvoří 12 praktik, které se navzájem doplňují a podporují
- Výhodou je její popularita - existuje dostatek literatury (i v českém jazyce)
- Nevýhodou metodiky je její velká striktnost, co se týká pravidel. Metodika by se měla implementovat celá, což často nebývá možné

XP – životní cyklus



XP – životní cyklus

- Explrorační fáze (Exploration Phase)
  - Začátek projektu
  - Zákazník sepíše možnosti použití softwaru (user stories) na karty zadání (story cards), které by rád měl v první verzi
  - Délka trvání iterace - mezi několika týdny a několika měsíci.
- Plánovací fáze (Planning Phase)
  - Zákazník dává priority jednotlivým požadavkům
  - Programátoři odhadnou jak dlouho tak může implementace jednotlivých karet zadání trvat
  - Pár karet zadání se vybere do první iterace
  - Obvykle zabere jen několik dní.
- Iterační fáze (Iterations to Release Phase)
  - Následuje po plánovací fázi a trvá až do vydání výsledku
  - První iterace má za úkol stanovit hrubou architekturu celého systému
  - Na začátku každé iterace zákazník vybírá, co se bude implementovat
  - Na konci iterace výsledky zákazník otestuje.

XP – životní cyklus

- **Produkční fáze (Productionizing Phase)**
  - Začíná tehdy, pokud se iteracemi povedlo vytvořit první produkt (Release)
  - V této fázi se nachází další testování a ověření možnosti nasazení systému u zákazníka.
- **Fáze údržby (Maintenance Phase)**
  - Především zajišťuje to, že udržuje systém u zákazníka v provozu
  - Zároveň vytváří další iterace pro další produkty
- **Smrt projektu (Death Phase)**
  - Nastává poté, co zákazník již nemá žádné další požadavky
  - Zde se napíše finální verze dokumentace
  - Nejsou prováděny žádné změny na produktu

XP – proměnné (stupně volnosti) projektu

- Extrémní programování rozehrává už u vyjednávání o podmínkách dohody se zákazníkem zajímavou hru, jejímž cílem je určení čtyř základních proměnných projektu:
  - Kvalita
  - Čas
  - Náklady
  - Rozsah
- Zákazník si vybere tři proměnné, které chce určit a jednu nechá na týmu
- Zákazníka je přinucen se nad vývojem zamyslet a rozhodnout, které z proměnných jsou pro projekt prioritní
- Zároveň musí uvážit, že pokud bude mít na své tři proměnné přemrštěné požadavky, tak dodavatel nastaví tu jednu svou na úroveň, která už jím nemusí být akceptovatelná
- Nutí to tedy zákazníka udělat jakýsi rozumný kompromis
- Proměnné by se po dobu projektu neměly měnit

XP – role a odpovědnosti

- **Přiřazení odpovědností určitým osobám je důležité - XP proto definuje role a určuje u nich co budou mít na starost**

- **Programátor (Programmer)**
- **Zákazník (Customer)**
- **Tester (Tester)**
- **Stopař (Tracker)**
- **Kouč (Coach)**
- **Konzultant (Konsultant)**
- **Manažer (Manager, Big Boss)**

- **Stopař**
  - **Má za úkol především sbírání zpětné vazby a kontrolu nad průběhem jednotlivých iterací, především, co se týká požadované funkcionality a časových odhadů**
- **Kouč**
  - **Hlídá průběh vývoje, dodržování předepsaná metodika a zda se lidé neuchylují od dané cesty**
  - **Hlavním jeho úkolem je především to, aby mezi sebou lidé správně komunikovali**

XP – praktiky

- XP používá 12 základních praktik, které se dají rozdělit do tří základních skupin
  - **Business praktiky**
    - Plánování hry
    - Zákazník na pracovišti
    - Malé verze
    - Metafora
  - **Týmové praktiky**
    - Párové programování
    - Společné vlastnictví
    - Standardy pro psaní zdrojového kódu
    - Udržitelné tempo
  - **Programovací praktiky**
    - Neustálá integrace
    - Jednoduchý návrh
    - Refaktorizace
    - Testování

XP – business praktiky

- **Plánování hry**
  - Úzká spolupráce zákazníka a vývojového týmu nad sestavováním zadání (User Stories)
  - Zákazník rovnou získává zpětnou vazbu od programátorů, kteří rovnou odhadují jak moc budou dané požadavky časově náročné a rovnou se tím sestavuje plán
  - Na řadě je opět zákazník a musí se rozhodnout, co je pro něj důležité a co má být implementováno jako první
  - Jednotlivé požadavky jsou pak seskupeny do jednotlivých iterací podle důležitosti

XP – business praktiky

- **Zákazník na pracovišti**
  - Opět bod související s komunikací se zákazníkem
  - Vyžaduje se, aby byla jedna osoba reprezentující zákazníka po celou dobu vývoje přítomna u programátorů
  - Pro některé zákazníky však tento bod bývá poměrně komplikovaný
  - K vytvoření použitelného software je potřeba mít k dispozici skutečného zákazníka, čím není myšleno toho, kdo bude platit, nýbrž toho, kdo bude systém používat
  - Tato osoba musí mít pravomoc určovat požadavky a stanovovat priority

XP – business praktiky

- **Malé verze**
  - XP tvrdí, že je třeba vydávat nové verze tak často, jak to jen jde
  - Cílem je dostat přínosné funkce k zákazníkovi co nejdříve
  - Programátoři tak získají rychlejší zpětnou vazbu
  - Dodáváním malých a častých verzí se zákazník vtahuje do děje
  - Zákazník pak ví, jak projekt postupuje, a může rovnou reagovat
  - Důraz se klade na co nejkratší dodávky klidně denní, nejdéle každý měsíc
- **Metafora**
  - Je důležité, aby lidé v týmu dokázali jednoduše o systému komunikovat
  - Metafora poskytuje systém jmen a popisů systému, které jsou zapamatovatelná, jednoduché k pochopení a sdílené celým týmem
  - Nalezení společného jazyka mezi programátory a zákazníkem je rovněž velmi důležité pro hladký průběh projektu

XP – týmové praktiky

- **Párové programování**
  - Nejznámější praktikou, která vstoupila do povědomí programátorů a která se většinou všem vybaví u názvu extrémní programování je párové programování.
  - Tato praktika je krásně a jednoduše popsána krátkou větou:
  - „Two people write the code at one computer“ (Dva lidé píší kód u jednoho počítače)
  - Zatímco jeden píše kód, druhý přemýšlí o souvislostech a optimálnosti řešení
  - Průběžně tak probíhá určitá revize kódu
  - Programátoři si své role v páru střídají
  - Stejně tak se obměňují i páry

XP – týmové praktiky

- **Společné vlastnictví**
  - Tento bod je poměrně specifický pro extrémní programování
  - Většina ostatních metodik klade důraz na to, aby za konkrétní zdrojový kód byly odpovědné konkrétní osoby v týmu. Aby objekt vždy vlastnila jediná osoba, která by ho mohla upravovat
  - XP místo toho říká, že „Kdokoli může kdykoli změnit jakoukoli část kódu“. Snaží se tím zvýšit tzv. *truck faktor* – aby částem kódu rozumělo více lidí a ne pouze jediný, který nám z ničeho nic může odejít
  - Funkce se přeci jenom opírají o testy, takže i když se něco pokazí, hned bude vidět kde
  - Společné vlastnictví kódu je také základem pro párové programování a refaktorování

XP – týmové praktiky

- **Standardy pro psaní zdrojového kódu**
  - Nutnost vytvořit standardy psaní kódu
  - „Tým by měl hovořit prostřednictvím kódu“
  - Dodržování standardu je nutné pro efektivní práci v párech a pro refaktorování
- **Udržitelné Tempo**
  - Metodologie XP zastává názor, že pokud jsou programátoři unavení, pak dělají více chyb a produkují špatný kód
  - Je třeba nastavit takové tempo, které bude udržitelné po dlouhou dobu
  - Přesná hodnota je individuální, ale obecně se pohybuje okolo 40ti hodin týdně
  - Přesčasy jsou v XP spíše nežádoucím jevem a také znakem možných problémů projektu
  - Někdy je sice třeba zapracovat déle, ale XP má pro tyto situace jednoduché pravidlo: Nemůžete pracovat přesčas, pokud jste pracovali přesčas minulý týden

XP – programovací praktiky

- **Neustálá integrace**
  - Požaduje se, aby se veškeré změny integrovaly co možná nejdříve minimálně jednou denně, do výsledného produktu a aby bylo vidět co je již hotovo a zároveň se mohlo průběžně testovat a tak okamžitě zjistit pokud něco nefunguje
  - Reakce na chybu je tak mnohem rychlejší
- **Jednoduchý návrh**
  - XP tvrdí, že je pravděpodobné, že se požadavky na software stejně změní a proto je neefektivní plánovat dopředu funkčnost, které by mohla mít účel v budoucnu
  - Nikdy se totiž nedokážeme připravit na veškeré změny a mnoho práce strávíme nad něčím, co se třeba nikdy nevyužije
  - Pravidlo zní,vždy vytvářet ten nejjednodušší program, který splňuje aktuální požadavky

XP – programovací praktiky

- **Refaktorizace**
  - Programátoři vylepšují kvalitu kódu tím, že při své práci neustále kontrolují, zda by se nedal kód vylepšit či zjednodušit
  - Eliminují se duplicity a zvyšují čitelnost
  - Díky společnému vlastnictví kódu může každý programátor bez obav vylepšovat jakýkoliv kód
  - Programátoři se mohou pouštět i do složitých a nebezpečných změn, protože jsou jištěni sadou testů
  - Kent Beck upozorňuje, že je důležité si uvědomit, že při refaktorování by jsme neměli do kódu implementovat novou funkčnost, pouze dělat úpravy ve stávající

XP – programovací praktiky

- **Testování**
  - Na rozdíl od většiny klasických metodik nemá testování v XP explicitně určenou fázi
  - XP používá především tzv. Unit testy
  - Požaduje se, aby funkční testy připravoval zákazník – ten by přece měl vědět nejlépe co potřebuje a vyžaduje
  - Programátoři v XP začínají vždy psaním testů a až následně vlastního funkčního kódu, který naplňuje požadavky dané testy
  - V XP musí kód procházet testy na 100%

XP – výhody a nevýhody

- **Výhody**
  - Velkou výhodou XP je to, že dbá na programátory, aby se jim lépe pracovalo a aby dobře mezi sebou komunikovali
  - XP nezatěžuje členy týmu zbytečnou byrokratickou zátěží, nedefinuje konkrétně žádné typy dokumentů, které by se měly vytvářet a vše ponechává na komunikaci a dohodě.
  - Mnoho z praktik programátoři rádi vítají, protože se jim zdají intuitivní
- **Nevýhody**
  - XP není všelék – hodí se na menší a střední projekty, u velkých již může docházet k chaosu
  - Kent Beck říká, že ideální velikost týmu by měla být mezi 3 a maximálně 20 pracovníky
  - Zavádění Extrémního programování také není úplně jednoduché

CMMI – historie vzniku

- CMMI – Capability Maturity Model Integration (Stupňovitý model zralosti – volný překlad)
- Historie vzniku modelu
  - Autor modelu – SEI-CMU (Carnegie Mellon University, Software Engineering Institute, Pittsburgh)
  - Ve spolupráci s Ministerstvem obrany USA
  - CMM - vyvíjen v letech 1987-1997
  - 2002 – CMMI version 1.1
  - 2006 – CMMI version 1.2

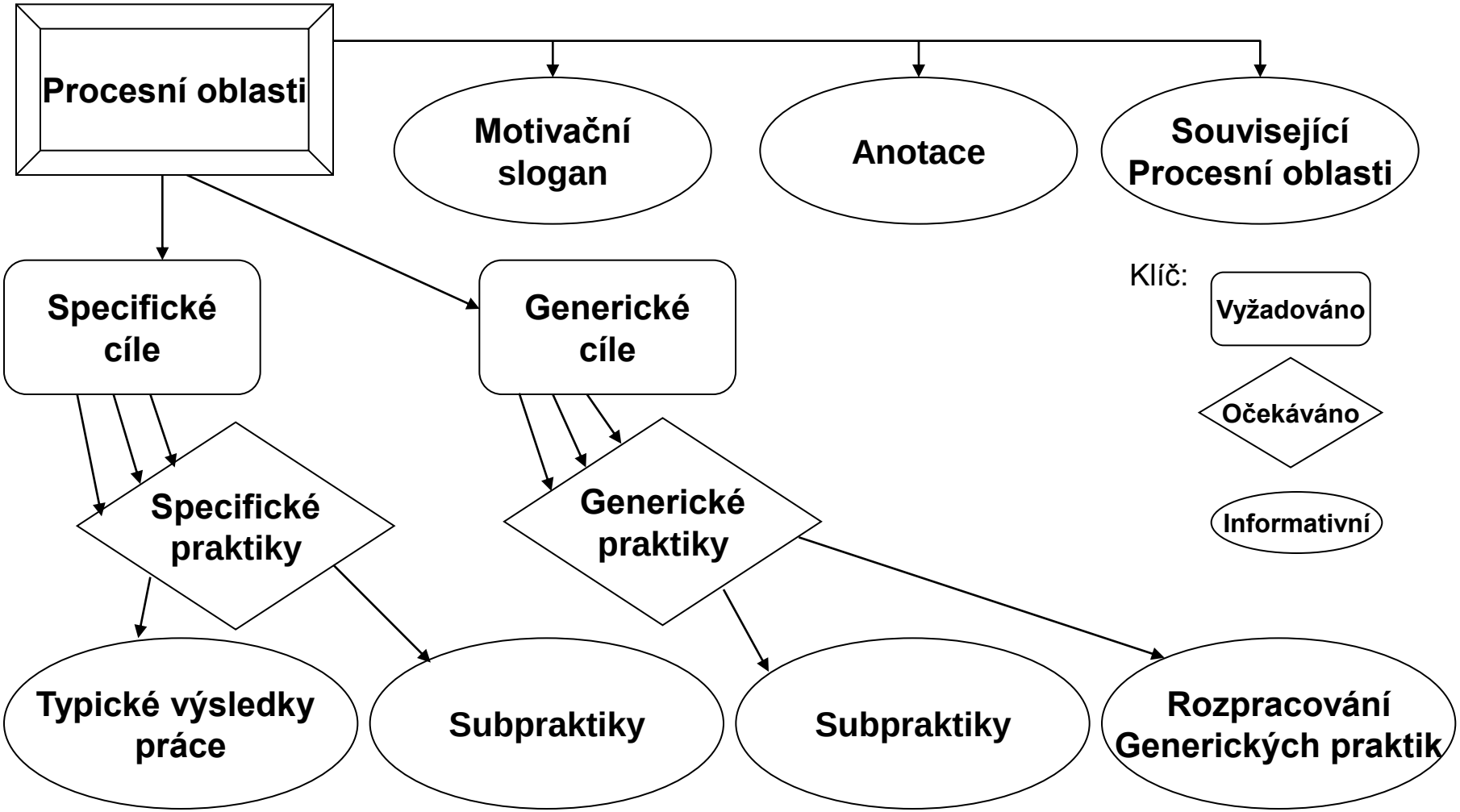
CMMI

- CMMI
  - Model kvality organizace práce určený pro vývojové týmy
  - Je určen pro vývoj produktů a služeb
  - Může být použit jako vodítko pro zlepšování procesů pro jednotlivé projekty, divize nebo celé organizace
  - Definuje procesní oblasti, které musí tým a organizace realizovat a cíle, které musí v každé oblasti splňovat
  - Je model zralosti procesů organizace
  - Je systém pro spolehlivé a důsledné hodnocení organizace
  - Je mechanismus sloužící k identifikaci a zavadění “best practices“

CMMI

- CMMI není
  - Návrh řešení
  - Standard
  - Technologie
  - Jazyk
  - Metodologie

CMMI – struktura modelu

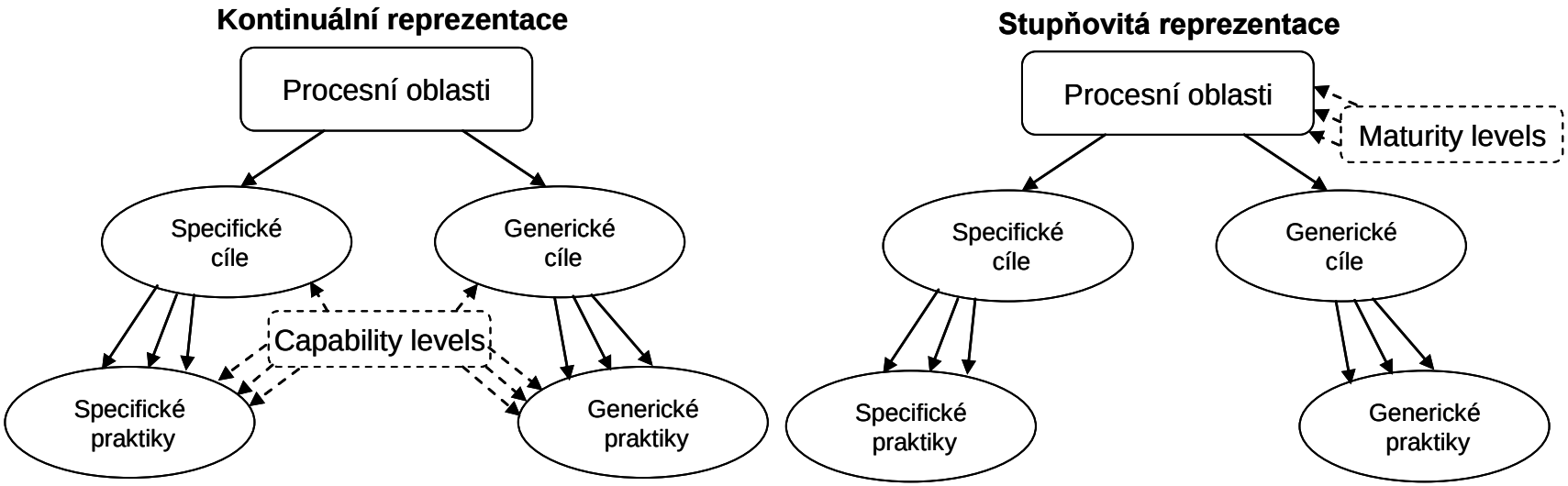


CMMI - terminologie

- **Specifické cíle (Specific Goals - SG)**
  - Specifický cíl je jedinečná charakteristika, která musí být přítomna, aby byla splněna příslušná procesní oblast.
- **Specifické praktiky (Specific Practises - SP)**
  - Specifická praktika je popis aktivity, která je považována za důležitou k tomu, aby byl dosažen odpovídající specifický cíl.
- **Generické (obecně použitelné) cíle (Generic Goals - GG)**
  - Generické cíle se nazývají “generické”, protože stejný cíl je použitý pro více procesních obastí.
- **Generické (obecně použitelné) praktiky (Generic Practices - GP)**
  - Generická praktika je popis aktivity, která je považována za důležitou k tomu, aby byl dosažen odpovídající generický cíl

CMMI - dvě reprezentace modelu

- CMMI umožňuje uživateli přistupovat ke zlepšování procesů a následnému hodnocení zlepšení dvěma různými způsoby díky dvěma rozdílným reprezentacím modelu:
  - continuous representation – kontinuální reprezentace
  - staged representation – stupňovitá reprezentace



CMMI - terminologie

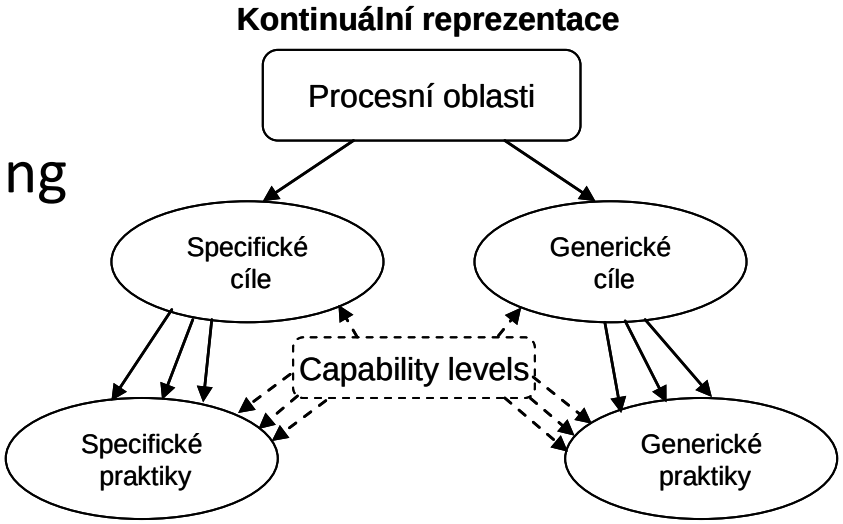
- **Specifické cíle (Specific Goals - SG)**
  - Specifický cíl je jedinečná charakteristika, která musí být přítomna, aby byla splněna příslušná procesní oblast.
- **Specifické praktiky (Specific Practises - SP)**
  - Specifická praktika je popis aktivity, která je považována za důležitou k tomu, aby byl dosažen odpovídající specifický cíl.
- **Generické (obecně použitelné) cíle (Generic Goals - GG)**
  - Generické cíle se nazývají "generické", protože stejný cíl je použitý pro více procesních obastí.
- **Generické (obecně použitelné) praktiky (Generic Practices - GP)**
  - Generická praktika je popis aktivity, která je považována za důležitou k tomu, aby byl dosažen odpovídající generický cíl.

CMMI - kontinuální reprezentace

- Maximální flexibilita – použití na zlepšování procesů
- Organizace si sama zvolí jednotlivý proces nebo několik procesů současně
- Umožňuje organizaci zlepšit různé procesy a dosáhnout různé úrovně zlepšení
- Existují jistá omezení ve výběru oblastí nebo procesů v důsledku závislostí mezi procesními oblastmi
- Měřítkem vyspělosti je „Capability level“
- 6 úrovní „Capability level“ – 0 až 5

CMMI - kontinuální reprezentace

- Level 0 – neúplná (chaos) / incomplete
- Level 1 – vykonávaná / performed
- Level 2 – řízená / managed
- Level 3 – definovaná / defined
- Level 4 – kvantitativně řízená / quantitatively managed
- Level 5 – optimalizující / optimizing



CMMI - stupňovitá reprezentace

- Nabízí systematičnost
- Zabezpečuje odpovídající infrastrukturu organizace
- Měřítko vyspělosti – „Maturity level“
- Dosažení jisté úrovně „Maturity level“ – dobré východisko pro dosažení úrovně vyšší
- Ukazuje cestu ke zlepšování organizace
- 5 úrovní „Maturity level“ – 1 až 5

CMMI – stupňovitá reprezentace

- **Level 1** – počáteční / initial
  - Procesy jsou obvykle chaotické a ad-hoc, organizace neposkytuje stabilní prostředí, které by podporovalo procesy. Úspěch v takovéto organizaci závisí na hrdinství zaměstnanců
- **Level 2** – řízená / managed
  - Procesy jsou plánovány a vykonávány v souladu s předpisy organizace. Procesy jsou navrženy spíše pro projekty a jsou reaktivní
- **Level 3** – definovaná / defined
  - Procesy jsou dobře definovány a chápany, jsou zakotveny v organizaci a jsou proaktivní
- **Level 4** – kvantitativně řízená / quantitatively managed
  - Procesy jsou měřeny a řízeny, proaktivní přístup
- **Level 5**– otpimalizující / optimizing
  - Procesy jsou neustále zlepšovány

CMMI – procesní oblasti

- Čtyři skupiny procesních oblastí

- Skupina **Řízení procesů**
  - Zaměření na procesy organizace
  - Definice procesů organizace
  - Školení organizace
  - Procesní výkonnost organizace
  - Zavádění novinek (inovace) a jejich rozšíření v rámci organizace
- Skupina **Řízení projektů**
  - Plánování projektů
  - Monitorování a řízení projektů
  - Řízení vztahů se subdodavateli
  - Integrované řízení projektů
  - Řízení rizik
  - Kvantitativní řízení projektů
- Skupina **Návrh a realizace (Engineering)**
  - Řízení požadavků
  - Vývoj požadavků
  - Technické řešení
  - Integrace produktu
  - Verifikace
  - Validace
- Skupina **Podpůrné procesy**
  - Řízení konfigurací
  - Zajištění jakosti produktů a procesů
  - Měření a analýza
  - Rozhodování na základě analýzy variant
  - Kauzální (příčinné) rozhodování na základě analýzy variant

CMMI – konfigurační management

- Účel konfiguračního managementu (CM) je vytvořit a udržovat integritu výsledků práce.
- Proces konfiguračního managementu zahrnuje:
  - Identifikaci zvolených výsledků práce, které tvoří “baselines” v daném časovém okamžiku
  - Řízení změn konfiguračních položek
  - Udržování integrity “baseline”
  - Vytvoření nebo poskytování specifikací, které vedou k vytvoření produktu na základě informací ze systému konfiguračního managementu
  - Poskytování přesného stavu a aktuálních konfiguračních dat vývojářům, uživatelům a zákazníkům

CMMI – konfigurační management

- Výsledky práce uložené pod konfigurační management zahrnují produkty dodávané zákazníkovi, produkty pro interní použití, nástroje, atd.
- Příklady výsledků práce, které mohou být uloženy pod konfigurační management:
  - Plány
  - Popis procesů organizace
  - Požadavky na produkt
  - Zdrojové kódy
  - Výkresy
  - Kompilátory
  - Testovací plány a procedury
  - Výsledky testů
  - atd.

CMMI – konfigurační management – specifické cíle a praktiky

- SG 1 – Vytvořit “baselines”
  - SP 1.1 – Identifikovat konfiguračních položky – indentifikovat konfigurační položky, komponenty a odpovídající výsledky práce, které budou uloženy pod konfigurační management
  - SP 1.2 – Vytvořit systém konfiguračního managementu – vytvořit a udržovat systém konfiguračního managementu a managementu změn sloužící pro řízení výsledků práce. Systém konfiguračního managementu zahrnuje záznamová média, procedury a nástroje pro přístup do konfiguračního systému.
  - SP 1.3 – Vytvořit nebo releasovat “baselines” – vytvořit nebo releasovat “baselines” pro interní použití a pro dodávky zákazníkům. “Baseline” je sada specifikací nebo výsledků práce , které byly formálně revidovány a odsouhlaseny, slouží pak jako základ pro další vývoj nebo dodávky a může být změněna pouze na základě procedury řízení změn.

CMMI – konfigurační management – specifické cíle a praktiky

- SG 2 – Sledovat a řídit změny
  - SP 2.1 – Sledovat požadavky na změny – Sledovat požadavky na změny konfiguračních položek. Požadavky na změny se netýkají pouze nových nebo modifikovaných požadavků, ale také vad výsledků práce.
  - SP 2.2 – Řídit konfigurační položky – Řízení zahrnuje sledování konfigurace každé konfigurační položky, schvalování nové konfigurace, pokud je to nutné a updatovaní “baseline”

CMMI – konfigurační management – specifické cíle a praktiky

- SG 3 -Vytvořit integritu
  - SP 3.1 – Vytvořit záznamy konfiguračního managementu – vytvořit a udržovat záznamy popisující jednotlivé konfigurační položky. Typické aktivity: historie revizí konfiguračních položek, záznam změn, stav konfiguračních položek, rozdíly mezi “baselines”, atd.
  - SP 3.2 – Vykonávat konfigurační audity – vykonávat konfigurační audity za účelem udržování integrity konfiguračních “baselines”. Konfigurační audity potvrzují, že výsledné “baselines” a dokumentace odpovídají specifikovaným standardům a požadavkům

Diskuse

DĚKUJI ZA POZORNOST