EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Metody řízení projektů - cesta k efektivitě a úspěchu

Učební texty k semináři

Autoři:

Ing. Jaroslav Lepka (Freescale Semiconductor, Rožnov p. R.)

Datum:

26. 2. 2010

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií CZ.1.07/2.3.00/09.0031

TENTO STUDIJNÍ MATERIÁL JE SPOLUFINANCOVÁN EVROPSKÝM SOCIÁLNÍM FONDEM A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

OBSAH

1. 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 IPMA (International Project Management Association) a PMI (Project Management Institute), které vydávají publikace a poskytují certifikace. Jednou z nejkomplexnější a nejpoužívanější publikací popisující podrobně problematiku projektového managementu je „A guide to the Project Management Body of Knowledge" [1] vydaný PMI. Publikace je průběžně aktualizována. Projekt management je o použití znalostí, dovedností, nástrojů a technik k dosažení cíle – splnění požadavků projektu. [1]

1.1. Definice projektu

Definicí pojmu projekt existuje celá řada. Přestože formulace definic se různí, všechny vystihují stejnou podstatu.

PMBOK definuje projekt jako:

• „A project is a temporary endeavor undertaken to create an unique product, service or result" [1].

• „Projekt je dočasně vynaložené úsilí k vytvoření unikátního produktu, služby nebo výsledku."

Podobnou definici má také norma ISO 10006 (Směrnice jakosti v managementu projektu)

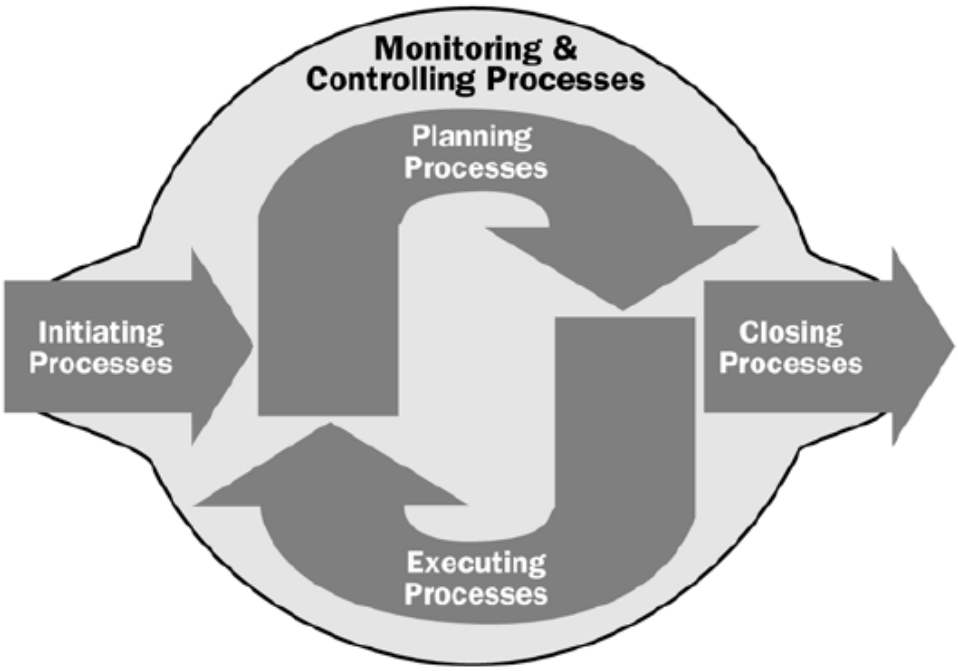
• „Projekt je jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení předem stanoveného cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji." [2]

1.2. Životní cyklus projektu (Project Life Cycle)

Životní cyklus projektu je doba od zahájení do ukončení projektu, která je rozdělena do čtyř fází: **iniciace**, **plánování**, **vykonávání** a **ukončení**. Skupiny procesů jsou mapovány do jednotlivých fází životního cyklu. Procesy každého projektu můžeme rozdělit do pěti skupin [1]:

- Iniciační/zahajovací procesy (Initiating Processes)
- Plánovací procesy (Planning Processes)
- Prováděcí procesy (Executing Processes)
- Monitorovací a kontrolní procesy (Monitoring & Controlling Processes)
- Ukončovací procesy (Closing Processes)

Obrázek 1.1 demonstruje jednotlivé procesní oblasti a jejich vzájemné propojení. Iniciační procesy projekt zahajují a ukončovací procesy naopak projekt a veškeré činnosti s tím spojené uzavírají. Plánovací a prováděcí fáze projektu se opakují v iteracích a vše je monitorováno a řízeno.



Obrázek 1.1 Životní cyklus projektu – procesní oblasti [1]

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.

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 – se zaměřuje 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 – je vyvrcholením projektového snažení, které má své náležitosti, jako je kontrola podle dohodnutých smluv, akceptace dodaných výsledků zákazníkem, závěrečná fakturace, atd.

1.3. Znalostní oblasti projektového managementu

PMBOK [1] dělí problematiku projektového managementu do devíti znalostních oblastí

1.3.1. *Oblast Integration Managementu*

Integration management 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.

1.3.2. *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.

1.3.3. *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.

1.3.4. *Oblast Cost Managementu*

Oblast cost managementu obsahuje procesy zapojené do plánování, odhadování, rozpočtování a kontroly nákladů, aby projekt byl dokončen s daným rozpočtem.

1.3.5. *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 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.

1.3.6. *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

1.3.7. *Oblast Communications Managementu*

Tato znalostní oblast 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) a jednání se zainteresovanými osobami k zajištění jejich požadavků.

1.3.8. *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

1.3.9. *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

Následující tabulka mapuje jednotlivé aktivity do znalostních oblastí a procesních skupin

Knowledge Area Processe	Process Groups				
	Initiating Process Group	**Planning Process Group**	**Executing Process Group**	**Monitoring & Controlling Process Group**	**Closing Process Group**
Project Management Integration	Develop Project Charter, Develop Preliminary Project Scope Statement	Develop Project Management Plan	Direct and Manage Project Execution	Monitor and Control Project Work, Integrated Change Control	Close Project
Project Scope Management		Scope Planning, Scope Definition		Scope Verification, Scope Control	
Project Time Management		Activity Definition, Activity Sequencing, Activity Resource Estimating, Activity Duration Estimating, Schedule Development		Schedule Control	
Project Cost Management		Cost Estimating, Cost Budgeting			Cost Control
Project Qyality Management		Quality Planning	Perform Quality Assurance	Perform Quality Control	
Project Human Resource Management		Human Resource Planning	Acquire Project Team, Develop Project Team	Manage Project Team	
Project Communication Management		Communicaton Planning	Information Distribution	Performance Reporting, Manage Stakeholders	

Project Risk Management		Risk Management Planning, Risk Identification, Qualitative Risk Analysis, Quantitative Risk Analysis, Risk Response Planning		Risk Monitoring and Control	
Project Procurement Management		Plan Purchases and Acquisitions, Plan Contracting	Request Seller Responses, Select Sellers	Contract Administration	Contract Closure

Tabulka 1.1 Mapování procesů do skupin a oblastí [1]

1.4. Plánování – tvorba časových plánů

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

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

Časový rozpis projektu představovaný diagramy a harmonogramy je významnou součástí plánu projektu a je dobrým nástrojem pro úplné a přehledné podchycení velkého kvanta informací potřebných pro řízení projektu. Diagramy a plánovací techniky prodělaly velký rozvoj v minulém století. Nedostatky jednoduchých tzv Ganttových diagramů a diagramů milníků, které neobsahovaly zobrazení závislosti mezi jednotlivými segmenty plánu, což komplikovalo posouzení dopadů na projekt v případě nějaké změny. Tyto problémy vyřešily síťové grafy. Samozřejmě s rozvojem výpočetní techniky a softwarové podpory dochází k modifikacím klasických metod, jako jsou Ganttovy diagramy a vlastnosti, které omezovaly jejich použití jsou implementovány.

Přehled metod založených 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)

1.4.1. *Ganttovy digramy*

Díky své jednoduchosti tvorby diagramu 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

Tabulka 1.2 Příklad Ganttova diagramu

V současnosti se Gantttovy diagramy hojně využívají pro svojí přehlednost, snadnou čitelnost a snadnou modifikovatelnost. Softwarové nástroje vedly ke zdokonalení Ganttových diagramů přidáním důležitých vlastností. Přibyly možnosti různých vazeb s překryvy a prodlevami, zobrazení milníků, zobrazení kritické cesty, zobrazení zdrojů a nástroje na porovnání odchylek mezi skutečným stavem projektu oproti plánu.

1.4.2. *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

Tabulka 1.3 Příklad tabulky milníků

1.4.3. *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. Zde se 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,* které lépe vystihuje proměnlivost provozních podmínek

Klíčový rozdíl v analýze PERT, na rozdíl od jiných metod analýz, je, že doba trvání každé činnosti je odvozena statisticky. To znamená, že trvání každé činnosti je určeno použitím tří časů:

- nejoptimističtějším odhadem
- nejpesimističtějším odhadem
- nejpravděpodobnějším odhadem

a potom použitím následujícího vzorce:

Te = **(a + 4m + b) / 6**

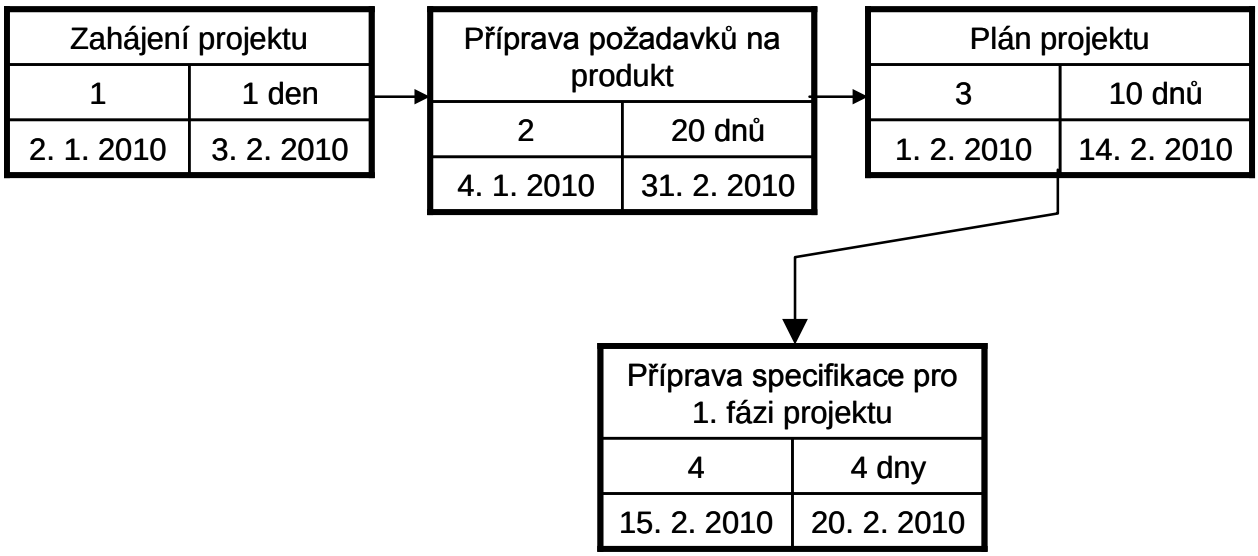
kde:

a - optimistický odhad

b - pesimistický odhad

m - nejpravděpodobnější odhad

Te - odhad doby trvání



Obrázek 1.2 Příklad PERT diagramu

1.5. Risk management

V okamžiku, kdy jsou detailně naplánovány všechny aktivity projektu a zdroje, nastává čas pro identifikování potenciálních rizik projektu a jejich ohodnocení. 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.

1.5.1. *Kvantifikace rizik*

Dalším krokem je kvantifikace pravděpodobnosti a dopadu příslušného rizika

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

Tabulka 1.4 Pravděpodobnost rizik

Pravděpodobnost (likelihood)

Doporučený způsob kvantifikace pravděpodobnosti rizik je znázorněn v Tabulka 1.4

Dopad / Vliv (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

Tabulka 1.5 Vliv rizik

2. AGILNÍ PŘÍSTUP K ŘÍZENÍ PROJEKTŮ

2.1. Důvody vzniku agilních metodik

Současná ekonomika je charakterizována neustálým tlakem na zkracování životního cyklu produktů. Důraz je kladen na inovace a rychlý technologický pokrok. Svět kolem nás se mění velmi rychle a být úspěšný znamená dokázat na změny nejen reagovat, ale dokonce je vyvolávat a tím získávat náskok před konkurencí. Současnému trendu přispívá rychlý rozvoj výpočetní techniky a software, jenž proniká do všech oblastí lidské činnosti. Překotné změny téměř ve všech odvětvích, do kterých významnou měrou zasahuje výpočetní technika vedly ke změně pohledu na řízení softwarových projektů. Současné softwarové projekty se vyznačují následujícími aspekty:

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

Zhruba od druhé poloviny 90 let minulého století se začínají objevovat snahy o nalezení nových metod řízení projektů, které by lépe odpovídaly novým požadavkům doby.

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“ (The Agile Manifesto) [3]. Tento dokument se stal základem rodiny agilních metodik a přístupů. Každá z agilních metodik je svým

způsobem specifická, ale všechny jsou postaveny na stejných principech, které byly definovány právě v Manifestu agilního vývoje softwaru. 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

2.2. Základní principy agilního přístupu vedení projektu

V dokumentu „The Agile Manifesto“ je definováno 12 základních principů, na kterých jsou založeny agilní metody řízení projektů

1) Naše nejvyšší priorita je uspokojit zákazníka pomocí brzkých a průběžných dodávek software, který přinese hodnotu.

2) Jsou vítány změny požadavků i v pozdějších fázích vývoje. Agilní procesy využívají změnu přinášející zákazníkovi konkurenční výhodu.

3) Časté dodávání fungujícího software v rozsahu od několika týdnů do několika měsíců s tím, že preferované jsou kratší časové úseky.

4) Obchodníci a vývojáři musí pracovat po celou dobu projektu společně každý den

5) Vytvářet projekty s okruhem motivovaných jednotlivců. Zabezpečit pro ně prostředí a podporu, kterou potřebují a věřit jim, že danou práci zvládnou.

6) Nejúčinnější a nejefektivnější metodou předávání informací v rámci vývojového týmu je osobní (phase-to-phase) komunikace.

7) Fungující software je hlavním/nejdůležitějším měřítkem pokroku projektu

8) Agilní procesy podněcují udržitelné tempo vývoje. Sponzoři, vývojáři a uživatelé by měli být schopni toto tempo udržet po neomezenou dobu

9) Neustálá pozornost směřovaná na technickou dokonalost a dobrý návrh zvyšuje agilitu (svižnost, hbitost)

10) Jednoduchost – umění maximalizovat množství práce, která se nemusí udělat je zásadní.

11) Týmy se samostatnou vnitřní organizací (samoorganizující se týmy) vynikají nejlepšími návrhy, architekturou i požadavky

12) V pravidelných intervalech by tým měl uvažovat nad tím, jak může být více efektivní a přizpůsobit tomu chování.

2.3. Agilní metodiky

Mezi agilní metodiky od počátku řadíme následující metodiky:

- 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)

2.4. Agilní metodiky, přístup k dokumentům, velikosti týmů a omezení agilních metodik

Obecně lze říci, že agilní metodiky nejsou rozpracovány do velkých podrobností (většinou nepředepisují tvoru masivní dokumentace, modely, atd.), kladou důraz na rychlý a efektivní vývoj zaměřený na spokojenost zákazníka, metodiky jsou iterativního charakteru (v podstatě je vývoj rozdělen do opakujících se bloků, kde zakomponování změn v průběhu projektu je relativně snadné),

výsledek projektu se vytváří inkrementálně, kdy jednotlivé inkrementy projektu lze použít před dokončením celého projektu. Speciální důraz je kladen na komunikaci se zákazníkem a uvnitř vývojového týmu. Takový způsob komunikace vede k lepšímu pochopení požadavků zákazníka, efektivnímu řízení a zakomponování změn, ke kterým dochází během vývoje projektu a v rámci týmu k efektivnímu předávání informací.

Agilní metodiky často chápou tvorbu a používání neoblíbených dokumentů jako zbytečné byrokratické postupy a staví dokumentaci na vedlejší kolej. Snaží se vyhnout tvorbě dokumentace pro dokumentaci, kterou často během projektu ani nikdo nečte a její tvorba jen zatěžuje vývojový tým. Podobně se přistupuje ke smlouvám, o kterých se vedou dlouhá jednání, jen aby si každý kryl záda nebo plánům, které se mění podle rozmaru zákazníka tak rychle, že jejich vytváření zabírá spoustu času.

Aby bylo možné agilní metodiky použít v praxi, musí být splněny některé základní předpoklady. Jeden z nejdůležitějších předpokladů je, aby zákazník byl k dispozici po celou dobu životního cyklu projektu a aktivně se podílel na vývoji. Dalším klíčovým předpokladem je velikost týmu a projektu, což spolu úzce souvisí. Obecně jsou agilní metodiky vhodné pro malé projekty a malé týmy, existují však i výjimky. Názory na velikost týmů se různí, běžně se jedná o zhruba 8 až 20 lidí, ale existují i příklady velikosti týmů v počtu 100 a více lidí. Třetím klíčovým požadavkem je kvalita vývojářů – jejich znalosti, zkušenosti i morální hodnoty [5].

Na základě požadavků, které musí být splněny pro použití agilních metodik, můžeme vyvodit omezení agilního vývoje [4]:

- omezená podpora pro distribuované prostředí vývoje
- omezená podpora subdodavatelů
- omezená podpora pro vytváření znovupoužitelných artefaktů
- omezená podpora pro vývoj ve velkém týmu
- omezená podpora pro vývoj kritických aplikací
- omezená podpora pro vývoj velkého komplexního softwaru

2.5. Nástroje pro podporu agilního vývoje

Agilní vývoj nutně nevyžaduje používání speciálních nástrojů. Lze používat agilní metody vývoje a nepoužívat jiný nástroj než běžný textový editor, sytém indexních karet pro zaznamenávání požadavků a knihovnu jednotkových testů. V posledních letech se na trhu objevilo několik nástrojů, které mnohem lépe podporují agilní vývoj. Hlavními dodavateli těchto nástrojů jsou Rally Software Development, VersionOne (produkt V1) and ThoughtWorks (produkt Mingle). Také IBM se svou rodinou nástrojů Rational nezůstává pozadu. Od roku 2006, kdy Scott Ambler, významný propagátor agilních přístupů, nastoupil do IBM, je vidět v rámci IBM významný důraz na agilní metody, který se projevil mimo jiné vznikem nové vývojové platformy Jazz. Také firma Microsoft podporuje agilní metody vývoje. Dodává metodiku Microsoft Solutions Framework (MSF), která je ovlivněna zejména myšlenkami Extrémního programování a zároveň zahrnuje praktiky používané při vývoji softwaru v samotné firmě Microsoft. MSF se dodává.ve dvou verzích:

- MSF for CMMI Process Improvement, která je více formální
- MSF for Agile Software Development

Velkou výhodou metodiky MSF je, že je přímo integrována do nástroje – Visual Studio Team System, a tak je „vynuceno“ její dodržování. Pokud vývojový tým preferuje přímo metodiku Scrum, může využít software Conchango (šablona pro metodiku Scrum), který lze volně stáhnout jako doplněk pro MS Visual Studio Team System.

2.6. Metodika Scrum

Metodika SCRUM Development Process (dále jen Scrum) byla představena Kenem Schwaberem a Mikem Beedlem v roce 2002 [6]. 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.

2.6.1. *Pojmy používané ve Scrum metodice*

Sprint

Sprint je první typ iterace, který se pravidelně opakuje. 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 *Porduct Backlogu*, které se budou implementovat v plánovaném *spritnu*. Výběr obvykle probíhá podle priorit, které definuje zákazník 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.

Backlog

V rámci metodiky *Scrum* existují dva druhy těchto *Backlogů*. Je to *Product Backlog* a *Sprint Backlog*. *Product Backlog* je nejdůležitějším dokumentem celé metodiky *Scrum*.

Tady jsou zaznamenány všechny požadavky zákazníka na software. Zákazník rovněž určuje prioritu pro každý požadavek. Je vhodné setřídit *Product Backlog* podle logických a funkčních celků, aby byl co nejvíce přehledný. Položky *Product Backlogu* jsou průběžně aktualizovány, přidávány a odstraňovány, aby vždy odpovídaly aktuálnímu stavu požadavků. Často používá seznam udržovaný jako tabulka v nějakém tabulkovém procesoru podobnému Microsoft Excelu.

Tým ve spolupráci se *Scrum Masterem* pak z *Product Backlogu* na základě priorit specifikovaných zákazníkem a logiky vývoje projektu vybere jednotlivé

položky a přesune je do *Sprint Backlogů*, které jsou pak implementovány v jednom sprintu.

Daily Scrum

Jde o druhý typ iterace, který je o mnoho kratší, trvá 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.

2.6.2. *Charakteristika*

Scrum byl vyvinut pro podporu řízení vývojového procesu. Rozhodně se nejedná o žádný 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, která se 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. To znamená, že je iterativní, flexibilní, dbá na rychlé dodávky částí aplikace nebo prototypů a následné sbírání zpětné vazby od zákazníka a snaží se rychle reagovat na měnící se požadavky a změny během vývoje.

2.6.3. *Role a odpovědnosti*

Metodika *Scrum* rozlišuje ve vývojovém týmu celkem šest rolí. Podobně jako v ostatních metodikách, jsou role důležité pro určování povinností a odpovědností. Následuje členění představené podle autorů metodiky [10].

Scrum Master

Je podobná funkce jako Kouč v Extrémním programování (viz. Extrémní programování) Je odpovědný za to, že proces vývoje probíhá v souladu s pravidly a praktikami metodiky *Scrum*. Komunikuje jak s vývojáři, tak také 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 odstranit sám, musí být schopen najít v týmu kompetentního a schopného člověka, který problému rozumí a je schopen ho odstranit.

Vlastník produktu (Product Owner)

Je zajímavá postava stojící někde na rozhraní mezi zákazníkem a dodavatelem. Je určován *Scrum Masterem*, zákazníkem a managementem. Jeho hlavní odpovědností je *Product Backlog*, odpovídá za úpravu a publikaci *Product Backlogu*. 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 sice často mění nebo přidávají, ale pouze mimo sprinty.

Scrum tým (Scrum Team)

Je hlavní vývojová síla. Jeho zodpovědnost není pouze v implementaci. Tým by měl fungovat samorganizujícím způsobem a jeho 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)

Zákazníci se především podílí na sestavování seznamu položek pro *Product Backlog*.

Management

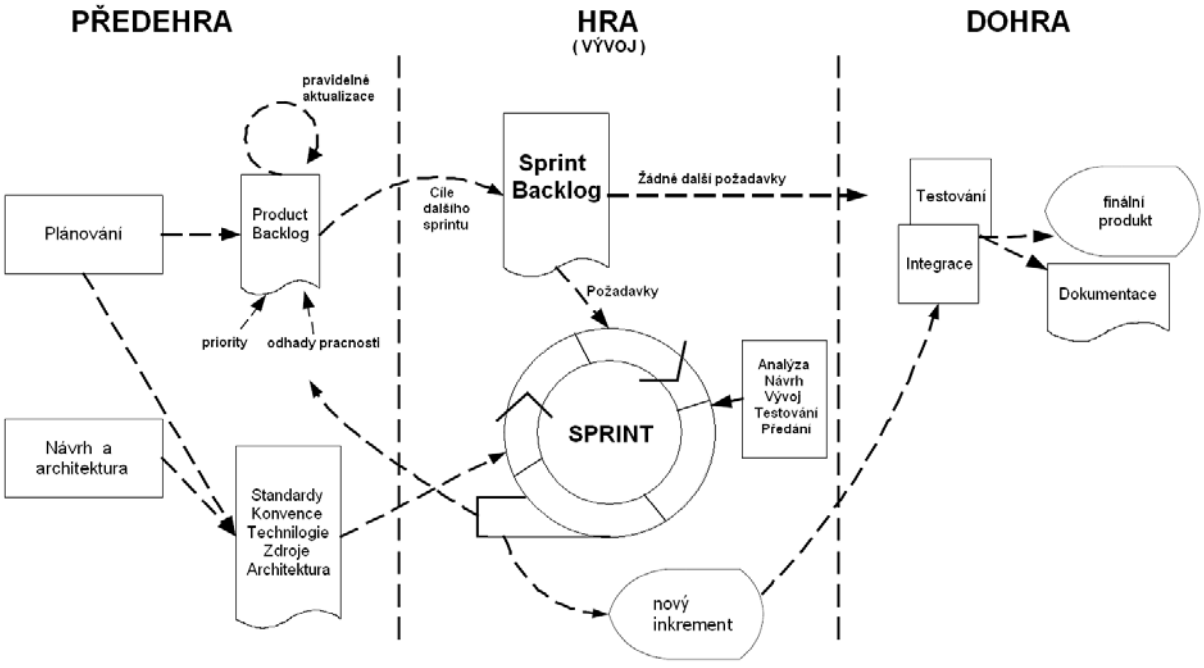
Management má na starosti především obchodní jednání a uzavírání smluv. Svoji roli plní také v případě důležitých rozhodnutí ovlivňujících celý proces. Podílí se na sestavování požadavků a cílů celého projektu i jednotlivých etap.

Uživatel (User)

Každý produkt je vytvářen pro někoho. Uživatel by měl zprostředkovávat zpětnou vazbu k jednotlivým inkrementům sprintů a k finálnímu produktu

2.6.4. *Fáze vývoje*

Scrum metodika dělí proces tvorby produktu do tří hlavních fází. Ty se příznačně jmenují **předehra**, **hra** a **dohra**. První a třetí fáze je lineární, prostřední fáze (hra) skládající se ze sprintů, se iterativně opakuje. *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.



Obrázek 2.1 Scrum – fáze vývoje

Obrázek 2.1, převzatý z [1] znázorňuje celý proces metodiky *Scrum*.

Předehra

Tato fáze se dělí do dvou podfází: **Plánování** a **Návrh a architektura**.

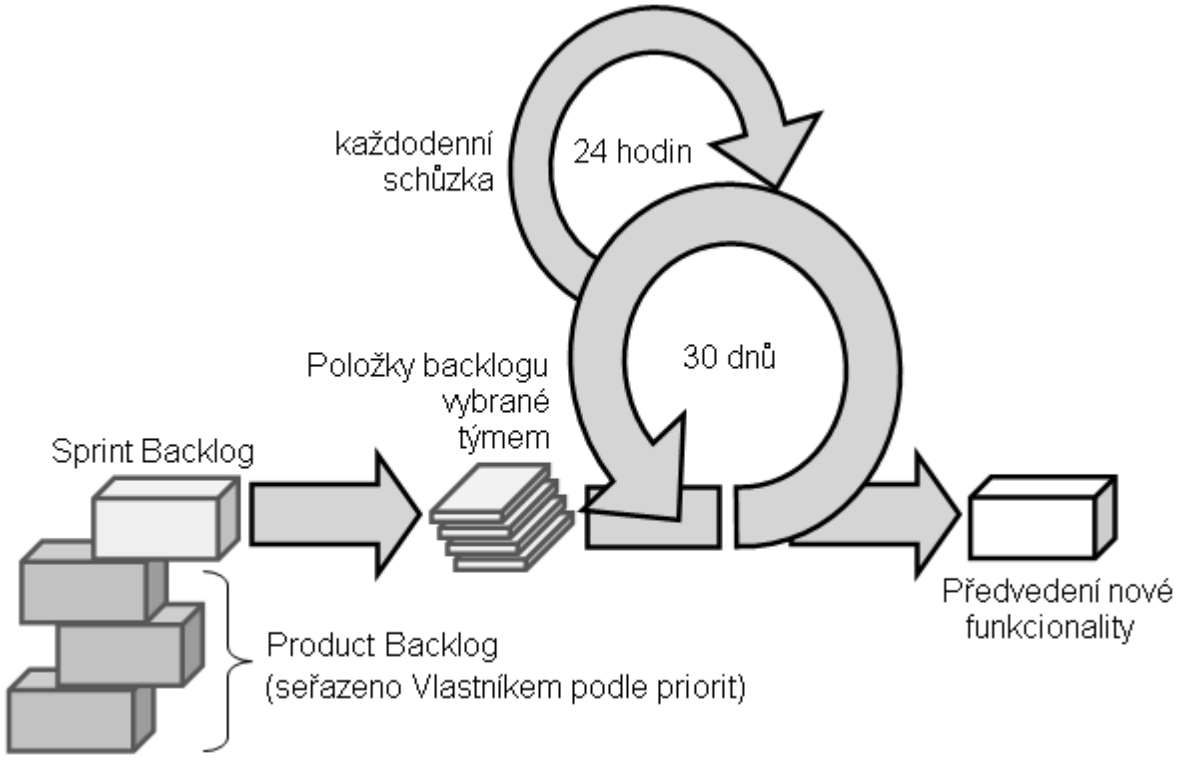
Plánování obsahuje definici systému, který má být vytvořen. Je sestaven *Product Backlog* obsahující všechny známé požadavky. Ty mohou přicházet od

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. Úkolem fáze Plánování je také sestavit projektový tým, definovat vývojové nástroje, analýzu rizik, zdroje a postup schválení výsledného produktu.

Návrh a architektura se zaměřuje na 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.

Hra

Je hlavní částí, na kterou se *Scrum* zaměřuje. V této fázi je vytvářen produkt. Tato fáze je hlavním zdrojem agility celého procesu *Scrum*. Přímo se říká:



Obrázek 2.2 Scrum iterace – představuje agilitu

Unpredictable is expected – Nepředvidatelné je očekáváno

Je členěna do tzv. *Sprintů*, kterých je obvykle tři až osm a každý Sprint trvá obvykle jeden měsíc. V rámci jednoho sprintu pak probíhají všechny klasické vývojové fáze: sbírání požadavků, analýza, návrh, vývoj, testování, předání. Každý sprint zpracovává část požadavků a funkčnosti zaznamenané v *Product Backlogu* a výsledek z každého sprintu je prezentován zákazníkovi. Veškerý vývoj je soustředěn do této fáze. Obrázek 2.2 ukazuje jednotlivé kroky, vstupy a výstupy v rámci jednoho sprintu.

Dohra

Fáze dohry nastává v okamžiku, kdy je kompletně zpracovaný celý *Product Backlog* a co je důležité, zákazník do něj již žádné požadavky nezadává a stávající požadavky nemodifikuje. V této fázi se vytváří finální release produktu. Provádí se integrace celého systému a probíhají závěrečné akceptační testy. Zajímavostí je, že dokumentace je vytvářena až v této fázi, v podstatě až na konci celého projektu.

2.6.5. *Praktiky*

Každodenní schůzka (Daily Scrum Meeting)

Tato aktivita je nejtypičtější 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 ovšem neřeší přímo

na schůzce, pouze se zaznamenají a jejich řešení má později po skončení schůzky na starost skupina lidí, kterých se problém týká a jsou schopni ho vyřešit.

Flexibilní předměty dodání (Flexible Deliverables)

Obsah dodávek není předem přesně dán, ale je závislý od typu projektu a prostředí, ve kterém nebo pro které se vyvíjí. Metodikou není striktně nařízeno, že výsledek analýzy musí být zpracován podle nějaké normy, protože v některých případech může být daleko užitečnější například objektový model nebo prototyp.

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.

Plánovací schůzka sprintu (Sprint Planning Meeting)

Tato schůzka je svolávána na začátku každého sprintu, organizuje ji *Scrum Master* a má dvě fáze. Té první 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*. V něm nemusí být pouhá podmnožina položek *Product Backlogu*, ale některé z nich mohou být detailněji rozpracovány do více položek. Druhé fáze se účastní jen *Scrum Master* a *Scrum tým* a zaměřují se na konkrétní implementaci daných položek.

Hodnotící schůzka sprintu (Sprint Review Meeting)

Svolává ji *Scrum Master* společně se *Scrum týmem* poslední den sprintu. Účastní se jí Vlastník produktu, management, zákazník a zástupci uživatelů. Právě na ní se zákazníkovi předávají výsledky sprintu. 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.

2.7. Extrémní programování (Extreme Programming)

Extrémní programování (XP) je dnes asi nejznámějším zastupitelem agilních metodik. 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“** [7]

Extrémní programování se tedy zaměřuje především na psaní zdrojového kódu.

Metodika Extrémní programování [8] je poměrně propracovaná metodika. Její jádro tvoří 12 praktik, které se navzájem doplňují a podporují [9]. Její výhodou je její popularita, díky níž existuje dostatek literatury (i v českém jazyce). Díky většímu počtu uživatelů lze na internetu nalézt i obsáhlá diskusní fóra, postřehy z praxe a lze tak obdržet radu i konzultaci. Nevýhodou metodiky je její velká striktnost, co se týká pravidel. Metodika by se měla implementovat celá, což často nebývá možné.

2.7.1. *Životní cyklus XP*

Autor extrémního programování – Kent Beck, definoval životní cyklus projektu následovně:

Na úvod projektu v explorační fázi zákazník sepíše možnosti použití softwaru (user stories) na karty zadání (story cards), které by rád měl v první verzi, délka trvání iterace se pohybuje mezi několika týdny a několika měsíci.

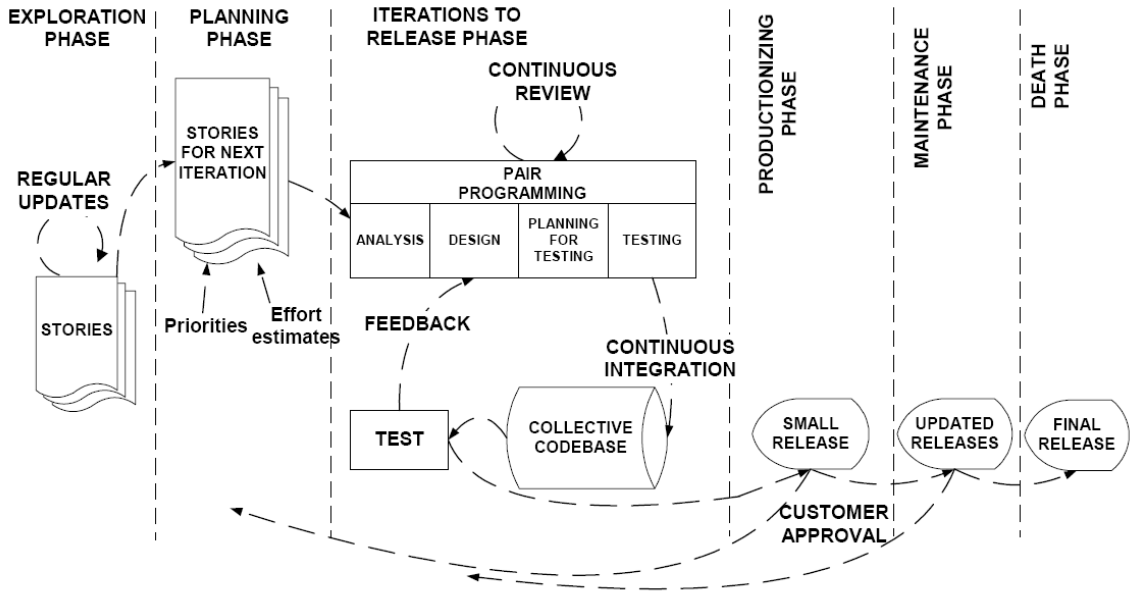
V plánovací fázi zákazník dává priority jednotlivým požadavkům, programátoři odhadnou jak dlouho tak může implementace jednotlivých karet zadání trvat. Pár karet zadání se vybere do první iterace. Fáze plánování obvykle zabere jen několik dní.

Poté následuje fáze iterací až do vydání výsledku, přičemž první iterace má za úkol stanovit hrubou architekturu celého systému. Na začátku každé iterace zákazník vybírá, co se bude implementovat a na konci iterace výsledky zákazník otestuje.

Produkční fáze 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 především zajišťuje to, že udržuje systém u zákazníka v provozu a zároveň vytváří další iterace pro další produkty.

Poté, co zákazník již nemá žádné další požadavky, nastává fáze smrti projektu (Death Phase), zde se napíše finální verze dokumentace a už nejsou prováděny žádné změny na produktu.



Obrázek 2.3 Vývoj podle XP

2.7.2. *Stupně volnosti projektu*

Extrémní programování rozehrává už u vyjednávání o podmínkách dohody se zákazníkem zajímavou hru. Jejím cílem je určení čtyř základních proměnných projektu, těmi jsou:

- Kvalita
- Čas
- Náklady
- Rozsah

Nechává zákazníka, aby si z těchto čtyř vybral tři, které chce určit, a aby jednu zbývající nechal na týmu. Toto nutí zákazníka se nad vývojem zamyslet a rozhodnout se, jaké z těchto proměnných jsou pro projekt prioritní, zároveň ale musí uvažovat nad tím, že pokud bude mít na své tři proměnné přemrštěné požadavky, tak jediné co může očekávat, že dodavatel nastaví tu jednu svou na úroveň, která už jím nemusí být akceptovatelná. Nutí to tedy zákazníka udělat jakýsi rozumný mix. Tímto se tyto proměnné nastaví pro zbytek projektu a už by se neměly měnit.

2.7.3. *Role a odpovědnosti*

Přiřazení odpovědností určitým osobám je důležité, XP proto definuje role a určuje u nich co budou mít na starost. Uvedu zde pouze jejich výčet, podrobnější popis můžeme nalézt například v [12] a jen krátce popíši některé role, které možná nebudou čtenáři tolik známé a v jiných metodikách se nemusí objevit.

- 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 – Další zajímavá role v XP, má hlídat průběh vývoje – tedy to, zda se dodržuje předepsaná metodika a lidé se neuchylují od dané cesty. Kouč by se dal se přirovnat ke SCRUM Masterovi Hlavním jeho úkolem je především to, aby mezi sebou lidé správně komunikovali.

2.7.4. *Praktiky extrémního programování*

12 základních praktik používaných v extrémním programování se dá rozdělit do tří základních oblastí. Business praktiky, týmové praktiky a programovací praktiky

2.7.4.1. Business praktiky

Plánování hry

Jedná se o úzkou spolupráci zákazníka a vývojového nad sestavováním zadání (User Stories), kdy 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

Zákazník na pracovišti

Opět bod související s komunikací se zákazníkem. Vyžaduje přímo to, 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. Umožňuje lepší komunikaci mezi vývojáři a zákazníkem.

Malé verze

Extrémního programování tvrdí, že je třeba vydávat nové verze tak často, jak to jen jde. Tak aby se nové, přínosné funkce dostaly k zákazníkovi co nejdříve. Programátoři tak mají také rychlejší zpětnou vazbu. Dodáváním malých a častých verzí se zákazník vtahuje do děje. Ví pak, jak projekt postupuje, a může rovnou reagovat. Dbá se na to, aby tyto dodání byly co možná nejkratší, klidně i denní, nejdéle však každý měsíc.

Metafora

V projektovém týmu je důležité, aby všichni dokázali jednoduše o systému komunikovat. Proto je zde metafora která poskytuje systém jmen a popisů systému, které jsou zapamatovatelná, jednoduché k pochopení a především sdílené celým týmem Nalezení společného jazyka mezi programátory a zákazníkem je velmi důležité pro hladký průběh projektu

2.7.4.2. Týmové praktiky

Párové programování

Nejznámější praktikou, která vstoupila do povědomí programátorů a která se většinou všem vybaví u názvu extrémní programování je párové programování. V [12] je tato praktika 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 přidávaného kódu, zda je to optimální řešení a podobně. Průběžně tak probíhá určitá revize kódu. Programátoři si své role v páru střídají. Stejně tak se obměňují i páry.

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

Standardy pro psaní zdrojového kódu

Pro dobrou spolupráci na společném kódu je třeba vytvořit určité společné standardy. Ať už se jedná o sjednocení odsazování či pojmenovávání, ve výsledku je třeba, aby byl kdokoliv schopný efektivně pracovat s kódem kohokoliv druhého. Hovoří se občas o tom, že tým spolu hovoří prostřednictvím kódu. Dodržování tohoto principu 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. Proto je třeba nastavit takové tempo, které bude udržitelné po dlouhou dobu. Přesná hodnota je individuální na konkrétním pracovníkovi 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

2.7.4.3. 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

Refaktorizace

Programátoři vylepšují kvalitu kódu tím, že při své práci neustále kontrolují, zda by se nedal kód vylepšit či zjednodušit. Eliminují duplicity a zvyšují čitelnost. Díky společnému vlastnictví kódu může každý programátor bez obav vylepšovat jakýkoliv kód. Díky společným standardům mu nebude dělat potíže vyznat se v cizím kódu a jeho nový kód bude také srozumitelný. Navíc se mohou programátoři pouštět i do složitých a nebezpečných změn, protože jsou jištěni sadou testů. Kent Beck upozorňuje, že je důležité si uvědomit, že při refaktorování bychom neměli do kódu implementovat novou funkčnost, pouze dělat úpravy ve stávající.

Testování

Na rozdíl od většiny klasických těžkých metodik nemá testování v XP explicitně určenou fázi. XP používá především tzv. Unit testy, požaduje, 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 zásadně procházet testy na 100%

2.7.5. *Výhody*

Velkou výhodou Extrémního programování je to, že dbá také 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. Možná právě díky tomu se Extrémní programování stalo velmi populární metodikou a dočkalo se poměrně dobré dokumentace a širokého spektra rozšíření. Mnoho z praktik programátoři rádi vítají, protože se jim zdají intuitivní.

2.7.6. *Nevýhody*

Extrémní programování však není všelék – XP je vhodné na menší a střední projekty, u velkých však již může docházet k chaosu. Kent Beck [7] říká, že ideální velikost týmu by se měla pohybovat mezi třemi a maximálně dvaceti pracovníky. Zavádění Extrémního programování také není úplně jednoduché.

Ač jsou třeba používané praktiky intuitivní, lidé si těžko zvykají na změny, a ty jsou v případě zavádění XP často velké. Najednou po nich nejsou požadovány rozsáhlé komplexní návrhy, dopodrobna definované plány, ale spíše se po nich chce, aby uvažovali jednoduše a byly odvážní dělat změny

3. CMMI (Capability Maturity Model Integration)

CMMI je model kvality organizace práce určený pro vývojové týmy a celé organizace. CMMI [13], zkratka anglického *Capability Maturity Model Integration*, se dá volně přeložit jako Stupňovitý model zralosti. Autorem modelu je Carnegie Mellon University, Software Engineering Institute, Pittsburgh (SEI-CMU) ve spolupráci s Ministerstvem obrany USA. CMM a CMMI modely byly vyvíjeny postupně, přičemž model CMM se vyvíjel v letech 1987 – 1997. V roce 2002 byl vydaný model CMMI verze 1.1 a v roce 2006 pak model CMMI verze 1.2. Model je určen pro vývoj produktů a služeb. Principy popsané v modelu mohou být použity jako vodítko pro zlepšování procesů pro jednotlivé projekty, divize nebo celé organizace. V rámci modelu se definují procesní oblasti, které musí tým a organizace realizovat a cíle, které musí v každé oblasti splňovat. Různá odvětví průmyslu zahrnující letectví, kosmonautiku, bankovnictví, software, hardware počítačů, telekomunikace, obranu, automobilový průmysl a mnoho dalších s úspěchem využívá CMMI pro vývoj.

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

3.1. Kontinuální reprezentace

Kontinuální reprezentace nabízí maximální flexibilitu, pokud chceme použít CMMI model pro zlepšování procesů. Organizace si pro zlepšení může zvolit jednotlivý proces nebo může pracovat na zlepšení v několika oblastech současně. Tato reprezentace rovněž umožňuje organizaci zlepšit různé procesy a v nich dosáhnout různých úrovní zlepšení, kde měřítkem vyspělosti je tzv. „Capability level“. Jistá omezení ve výběru oblastí nebo jednotlivých procesů jsou daná závislostmi mezi procesními oblasti.

Kontinuální reprezentace poskytuje 6 úrovní „Capability level“ v rozsahu 0 až 5:

Level 0 (úroveň 0) – neúplná (chaos) / incomplete

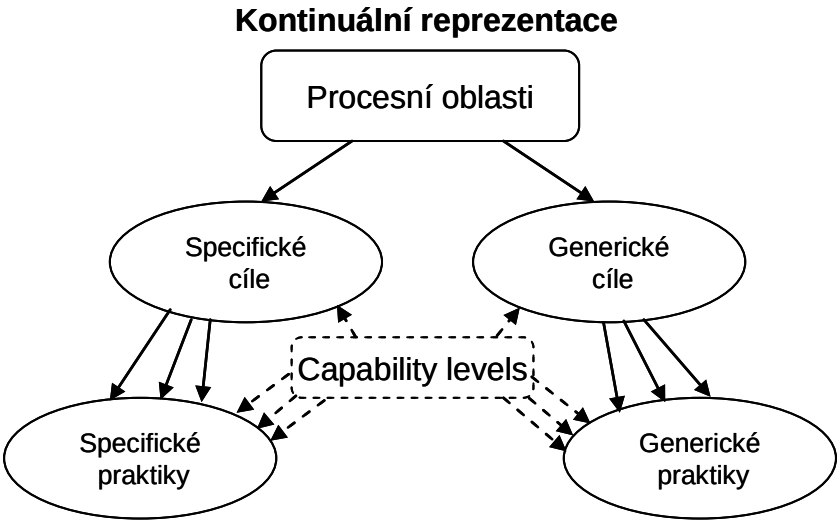
Level 1 (úroveň 1) – vykonávaná / performed

Level 2 (úroveň 2) – řízená / managed

Level 3 (úroveň 3) – definovaná / defined

Level 4 (úroveň 4) – kvantitativně řízená / quantitatively managed

Level 5 (úroveň 5) – otpimalizující / optimizing



Obrázek 3.1 Struktura kontinuální reprezentace

3.2. Stupňovitá reprezentace

Zatímco kontinuální reprezentace nabízí maximální flexibilitu ve zlepšování organizace, stupňovitá reprezentace nabízí systematičnost. Dosažení jistého stupně je ohodnoceno měřítkem vyspělosti tzv. „Maturity level“ a zabezpečuje odpovídající infrastrukturu procesů v rámci organizace. Rovněž dosažení jisté úrovně „Maturity level“ zajišťuje dobré východisko pro dosažení úrovně vyšší. Stupňovitá reprezentace ukazuje cestu ke zlepšování organizace.

Stupňovitý model poskytuje 5 úrovní „Maturity level“ v rozsahu 1 až 5:

Level 1 (úroveň 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 (úroveň 2) – řízená / managed

Procesy jsou plánovány a vykonávány v souladu s předpisy organizace. Procesy jsou navrženy spíše pro projekty a jsou reaktivní

Level 3 (úroveň 3) – definovaná / defined

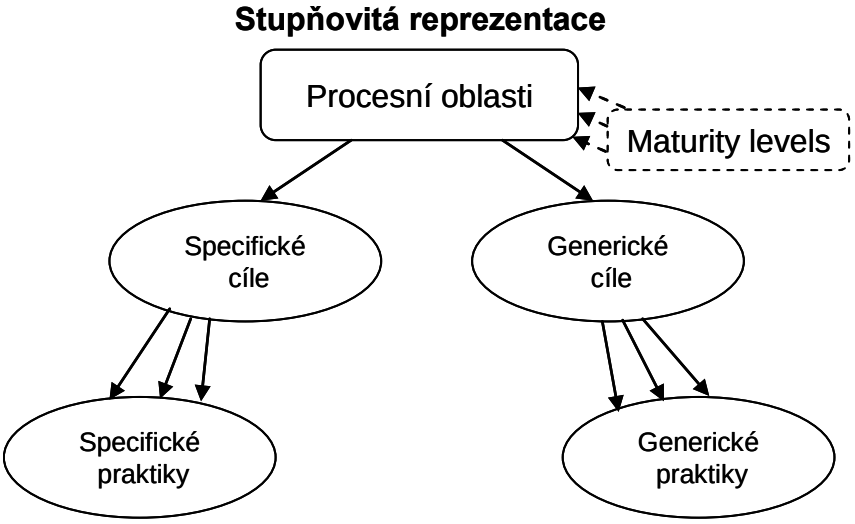
Procesy jsou dobře definovány a chápany, jsou zakotveny v organizaci a jsou proaktivní

Level 4 (úroveň 4) – kvantitativně řízená / quantitatively managed

Procesy jsou měřeny a řízeny, proaktivní přístup

Level 5 (úroveň 5) – otpimalizující / optimizing

Procesy jsou neustále zlepšovány



Obrázek 3.2 Struktura stypňovité reprezentace

3.3. Volba reprezentace modelu

V případě, že organizace není dobře obeznámena s jednotlivými reprezentacemi a je nováčkem v procesu zlepšování procesů, je jedno, kterou

reprezentaci si zvolí. V případě, že organizace již používá model CMM a má implementovánu konkrétní reprezentaci, doporučuje se v ní pokračovat, což umožní snadnější přechod od CMM k CMMI modelu.

3.4. Procesní oblasti modelu (kontinuální reprezentace)

CMMI model dělí procesní oblasti do čtyř skupin podle typu činností

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

3.5. Řízení konfigurací (Configuration Management - CM)

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

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.

3.5.1. *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.

3.5.2. *Přehled specifický cílů a praktik*

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.

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”.

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.

4. SEZNAM POUŽITÉ LITERATURY

[1] A Guide to the Project Management Body of Knowledge, Third Edition, 2004, Project Management Institute, Pennsylvania USA, ANSI/PMI 99-001-2004, ISBN 1-930699-45-X

[2] ČSN ISO 10006, Management jakosti – Směrnice jakosti v managementu projektu, Praha, 1999

[3] Agile manifesto. Dostupné na [www.agilemanifesto.org].

[4] Mullaney, J., Davidson, M. Software development trends in 2008, SearchSoftwareQuality.com

[5] Tur, D., France, R. and Rumpe, B. Limitations of Agile Software Process, http://www.agilealliance.org/system/article/file/1096/file.pdf

[6] Schwaber, Ken, Beedle, Mike. Agile Software Development with SCRUM, Prentice Hall 2002

[7] Václav Kadlec – Agilní programování, Metodiky efektivního vývoje softwaru, Computer Press, Brno, 2004, ISBN 80-251-0342-0

[8] Beck K.: *Extrémní programování* (český překlad) Grada 2002, ISBN 80-247-0300-9

[9] Pergl R.: *Analýza vnitřních vazeb principů metody extrémního programování*, In: Sborník příspěvků z doktorandského semináře, ČZU Praha, 2005, ISBN 80-213-1314-5

[10] Extreme programming approach, www.xprogramming.com

[11] Extreme Programming - a gentle introduction: http://www.extremeprogramming.org – jiná stránka o XP

[12] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen & Juhani Warsta – Agile software development methods – Review and analysis, VTT Electronics, University of Oulu, 2002, ISBN 951-38-6010-8 (URL: http://www.inf.vtt.fi/pdf/)

[13] Mary Beth Chrissis, Mike Konrad, Sandy Shrum, CMMI Second Edition – Guidelines for Process Integration and Product Improvement, version 1.2. Addison-Wesley, ISBN 0-321-27967-0

Centrum pro rozvoj výzkumu pokročilých řídicích a senzorických technologií
CZ.1.07/2.3.00/09.0031

Ústav automatizace a měřicí techniky
VUT v Brně
Kolejní 2906/4
612 00 Brno
Česká Republika

http://www.crr.vutbr.cz

info@crr.vutbtr.cz