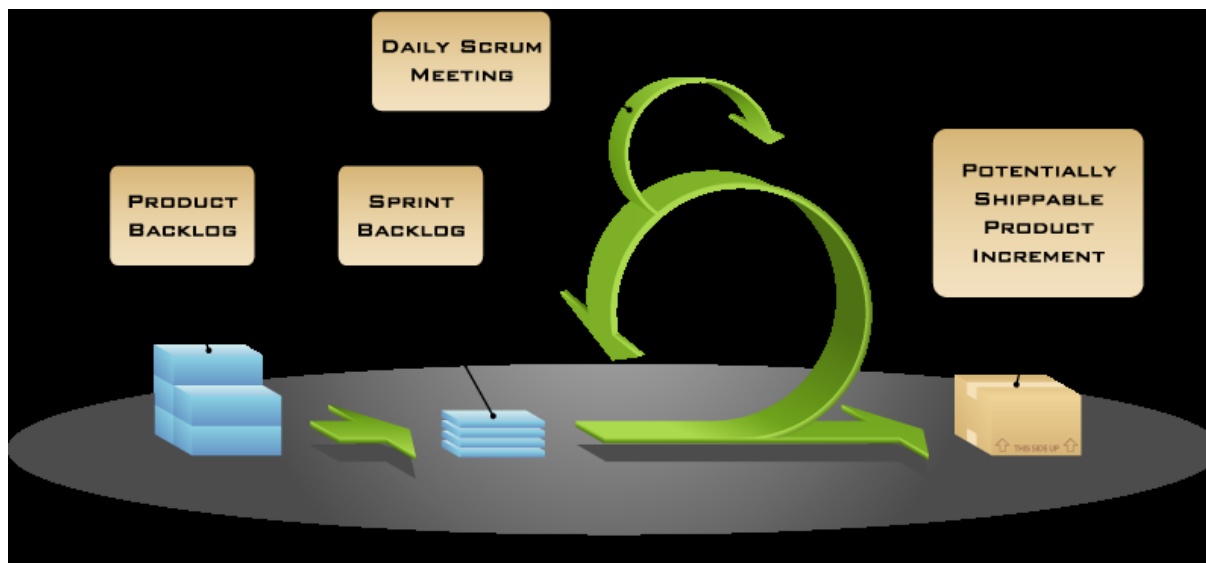
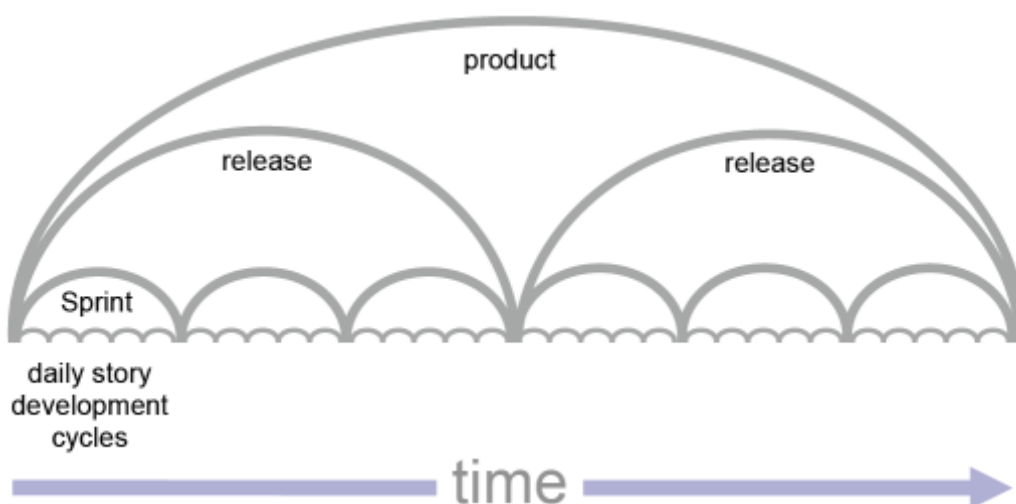


Přednáška č. 2 – Iterativní vývoj softwaru

SCRUM



Kontext iterace v procesu vývoje



Evoluční a adaptivní vývoj softwaru

- *evoluční*
 - znalosti o požadavcích, návrhu, odhadech a plánu se **vyvíjejí a zpřesňují v průběhu** projektu
 - míra změny většinou klesá s postupem projektu s iteracemi
- *adaptivní*
 - zdůraznění procesu učení
 - zpětná vazba od uživatelů
 - empirický proces

Charakter iterací dle fáze projektu

- základní schéma je pevné, mění se činnosti a artefakty
- *zahájení (inception)*
 - analytické činnosti, tvorba vize a validace zákazníkem (1-2 iterace)
- *projektování (elaboration)*
 - analytické a designérské činnosti, ověřování prototypy, implementace (2+ iterací)
- *konstrukce (construction)*

- designérské a programátorské činnosti, změnové řízení, testování a ověřování (N iterací)
- *nasazení (transition)*
 - integrační a konzultační činnosti, ověřování provozem, vytvoření uživatelské podpory (1-2 iterace)

Význam meziproduktů

- deskriptivní metodiky
 - artefakty jsou **cílem (výsledkem)** fáze (iterace) projektu
 - důsledek: review podpis →změnové řízení
- agilní přístup
 - artefakty jsou prostředkem (cíl = smysluplný stav, přírůstek produktu)
 - důsledky
 - forma, obsah artefaktů (dress code) pomocí šablon (RUP)
 - artefakty živé během celého projektu
 - výběr podle fáze a iterace

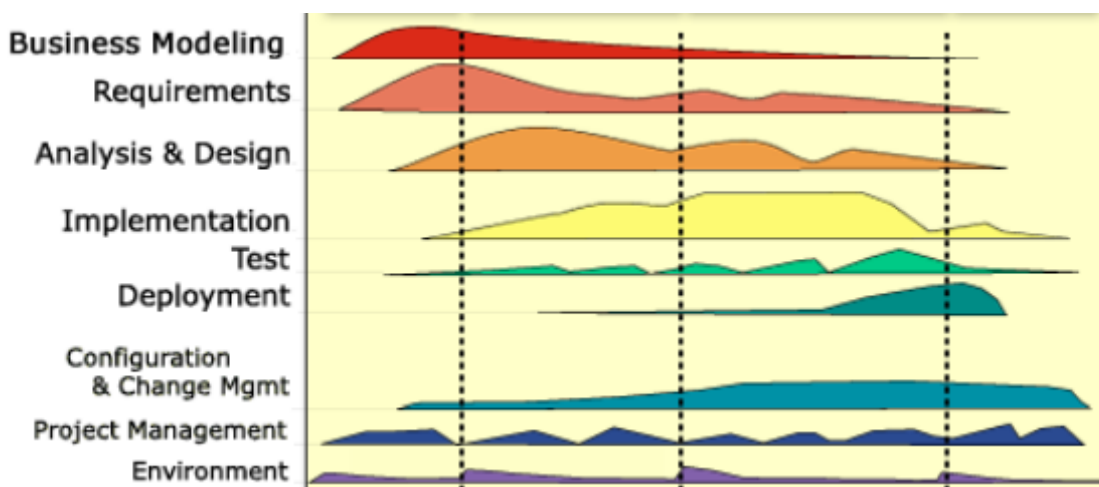
Přednáška č. 3 – Fáze Inception (zahájení)

Přehled a cíle

- nápad → nabídka → poptávka → formování vize (kontraktu) → zahájení projektu
- cíl fáze
 - vize produktu – co se má vytvořit
 - business case – vyplatí se projekt a proč
 - technický koncept – ověření proveditelnosti projektu

Klíčové disciplíny a aktivity

- rozsah a hranice projektu
- návrh ceny a harmonogramu práce
- klíčové funkcionality (případy užití – use cases) a omezující podmínky
- koncept technické architektury
- ověření proveditelnosti
- určení rizik projektu
- příprava prostředí projektu
- akceptační kritéria



Charakteristiky fáze

- důležitá je hlavně pro nově začínající projekty
- pro pokračující má význam znovu-ověření předpokladů a odhadů projektu

- počáteční chaos → důsledky pro plánování iterací
- význam business modelování a komunikace

Požadavky

- sběr požadavků a analýza (co se vlastně chce)
 - pochopení problému
 - model reálného/projektovaného světa
 - podklady – realizace a plán
- návrh, jak to realizovat
- úrovně detailu
 - zahájení projektu – klíčové, obrysy
 - projektování – podstatné, úplnost
 - konstrukce – podrobnosti

Lidé v analýze

- zákazník
 - externí, interní, doménový expert
- zainteresovaný hráč (stakeholder)
 - ředitel, investor, daňový poplatník, standardizační orgán
 - vliv na úspěch projektu
- analytik
 - zprostředkovatel mezi zákazníkem a programátory
 - co umí
 - komunikační schopnosti, naslouchání a pozorování
 - vedení schůzek, organizační
 - schopnost abstrakce, nadhled, tvořivost
 - detailní znalost problémové oblasti
 - psaný projev a schopnost modelování

Požadavky na software

- schopnost nebo vlastnost, kterou má program mít, aby jej uživatel mohl používat k vyřešení problému nebo dosažení cíle, který vedl k zadání tvorby programu, nebo aby splnil podmínky stanovené smlouvou, standardem, nebo jinou specifikací
- požadavky omezeny vnějšími podmínkami
- požadavkem není to, co uživatel nepotřebuje

Definice systému

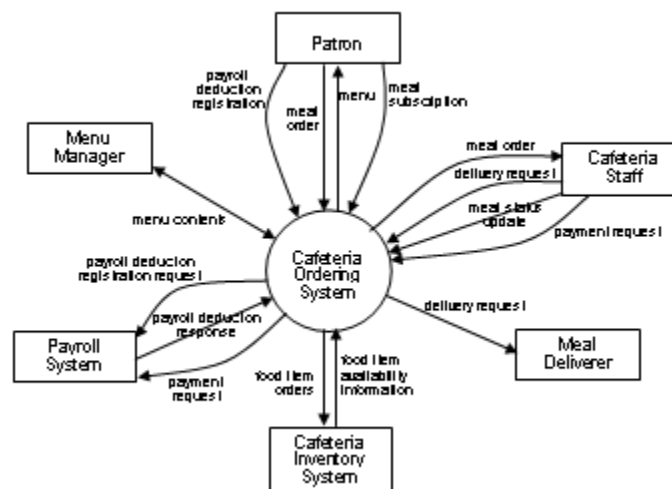
- základní a stručný popis účelu projektovaného systému
- vyjádření cílu projektu
 - soulad mezi zákazníkem a dodavatelem
 - zabránit divergování a změnám během vývoje
 - prevence nárůstu požadavků (feature creep)
- stupnice pro zhodnocení úspěšnosti projektu

Stručný popis problému

- 25 slov na 1 odstavec
 - k čemu má systém sloužit
 - jaké informace bude udržovat
 - kdo ho bude používat
 - co přinese a čemu pomůže
- Tisková zpráva
 - jak si představujeme výsledné řešení
- Šablona RUP
 - problém (co)
 - postihuje (koho)
 - vede to k (důsledky)
 - řešení je (cílový stav)⁷

Kontextový model

- Zobrazení stavu systému s ostatními (okolními) entitami
 - systém jako černá skříňka
 - aktéři / stakeholdeři
 - ostatní systémy
- Rozsah systému
- Rozhraní na okolí
 - HCI, API, data



Vize produktu a rozsah projektu

- popis cíle, který má vzniknout (výsledný produkt – popis)
 - klíčové, esenciální, vysokoúrovňové požadavky
 - problém za problémem
- související artefakty
 - business case
 - proč to vlastně chci
 - zdůvodnění výhodnosti – návratnosti projektu
 - požadavky
 - co to má vlastně dělat

Požadavky pomocí UML

- Popis požadavků na vnější funkčnost systému
 - jací uživatelé k němu přistupují
 - co pro ně systém dělá
 - jak systém zpracovává požadavky
 - kde je hranice systému (co vlastně řešíme)
- Pomůckou je 4+1 kategorie
 - **jaké informace systém obsahuje, udržuje**
 - **jaké funkce poskytuje uživatelům**
 - **jaké analýzy dat provádí**
 - **jaké jsou interakce s jinými**
 - pro každý druh příslušné vlastnosti (aspoň seznam)
 - +1 je not this time, až příště, doplňková funkčnost
- Modelem jsou aktéři a případy užití
 - kontext, primární funkčnost
 - další iterace, podružná funkčnost

Aktér

- uživatel nebo jiný systém, který náš systém používá nebo bude používat
 - typový uživatel – vně systému
 - spouští dané případy užití
 - primární a sekundární aktéři
- popis
 - název role, nikoli jména
 - jak používá systém a k čemu ho používá
 - seznam cílů
- vazby mezi aktéry jsou generalizace – hierarchie rolí
- jak je najít
 - odrážejí způsoby používání aplikace
 - jsou podstatní pro určení hranice systému
- vytřídit primární a sekundární
- uvnitř systému a not this time
- některé budou nalezeny až později

Případy užití

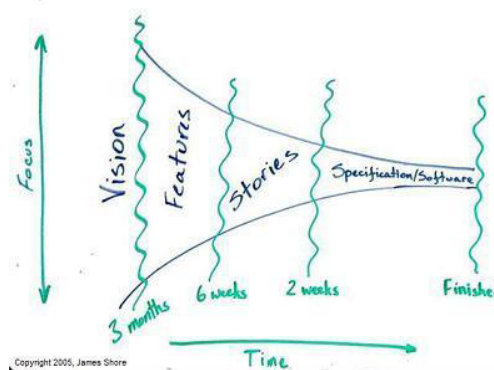
- sekvence akcí, které systém provádí na základě vnějšího podmětu (aktéra) a vedou k výsledkům viditelným pro některé aktéry
- jak je najít
 - dialogy aktér – systém
 - začíná se aktéry
 - popis problému od 1. aktéra
 - seznam cílů a potřeb – případ užití
 - hlavní akce, které provádí?
 - jak vkládá a získává informace?
 - potřebuje vědět o stavu systému?

Agilní specifikace požadavků

- cílem je říci, že produkt by měl umět něco
- ne funkce něco vypadá tak a tak
- podpořit (podnítit) budoucí diskuzi o detailech

Vývoj use stories

- počátky projektu a dosud neanalyzované oblasti
- (vize)
- feature
 - minimal marketable = repase level
 - pracnost > 3 iterace
- epic
 - větší funkcionalita
 - pracnost > 1 iterace
 - má akceptační testy
- user story
 - sada = theme



Glosář

- seznam důležitých pojmů
 - klíčové
 - nejasné
 - sporné
- stručný, všemi odsouhlasený popis
 - společný slovník
 - prevence nedorozumění
- formát různý
 - word, excel, DB

Doménový model

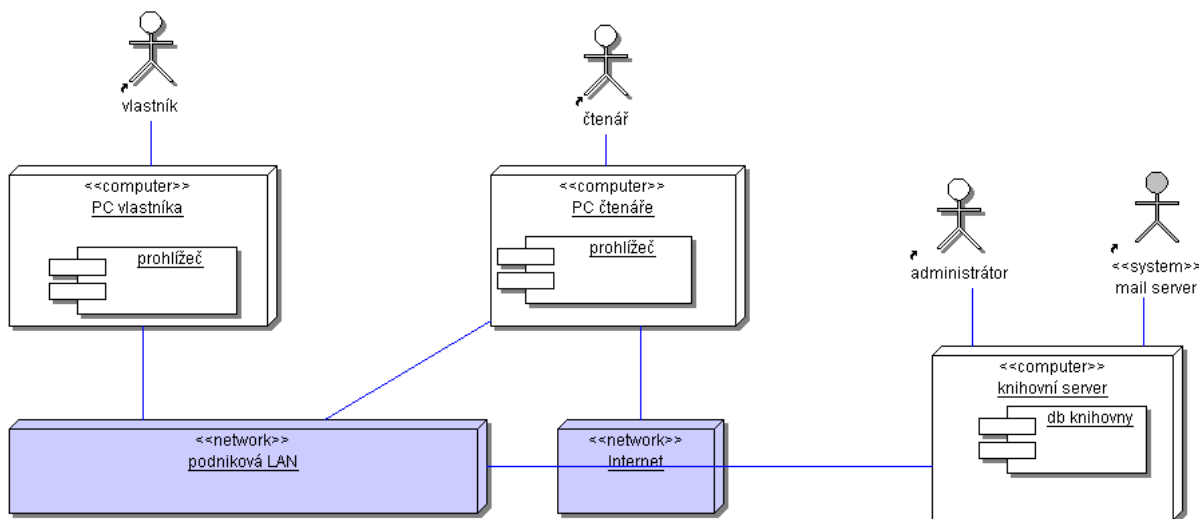
- popis struktury problémové oblasti
 - základní abstrakce používané v oblasti aplikace
 - názvy, vzájemné vztahy a vlastnosti
 - jakým postupem je získáme
 - podle čeho si máme vymyslet třídy pro realizaci
- východiskem je glosář
- model – doménové objekty – diagram tříd
- prvky
 - objekty – třídy

- věci, které se vyskytují v problémové doméně
- klíčové pro fungování systému
- systém udržuje informace
- podstatné aspekty
 - terminologie uživatele, pojmy → názvy tříd
 - jen základní obrysy
 - vztahy mezi třídami (vazby, kardinality)
 - nezávislost na implementaci
- jak najít doménové objekty
 - doménová analýza
 - konzultace
 - doménový expert
 - části systému podstatné z jejich pohledu
 - pomůžou
 - obrázky
 - pozorování práce
 - rozhovor s uživatelem

Fyzický rozsah systému

- Vztahy produktu a prostředí
 - porozumění run-time a fyzickému prostředí
 - charakteristiky a parametry – mimofunkční požadavky
 - odhad nákladů a podklad pro architekturu
- varianty
 - zelená louka – součást návrhu architektury
 - brownfield – nutná součást analýzy

Popis prostředí – diagram nasazení – mimofunkční charakteristiky



Způsoby získávání požadavků

- Neinteraktivní
 - studium dokumentace, hlášení problémů (bugtracking, support)
 - analýzy trhu
 - konkurenční systémy
- Interaktivní
 - rozhovory
 - pozorování, práce s uživateli

- marketingové průzkumy, dotazníky
- prototypování

Vývoj znalosti požadavků

Iterace	Část modelu	Hotovo
1	aktéři	<div style="width: 30%; background-color: #90EE90;"></div>
	seznam PU	<div style="width: 50%; background-color: #FFDAB9;"></div>
	popis PU	<div style="width: 20%; background-color: #ADD8E6;"></div>
	doménový model	<div style="width: 10%; background-color: #FFFF00;"></div>
	znalost požadavků	<div style="width: 10%; background-color: #8A2BE2;"></div>
2	aktéři	<div style="width: 60%; background-color: #90EE90;"></div>
	seznam PU	<div style="width: 80%; background-color: #FFDAB9;"></div>
	popis PU	<div style="width: 40%; background-color: #ADD8E6;"></div>
	doménový model	<div style="width: 60%; background-color: #FFFF00;"></div>
	znalost požadavků	<div style="width: 30%; background-color: #8A2BE2;"></div>
3	aktéři	<div style="width: 90%; background-color: #90EE90;"></div>
	seznam PU	<div style="width: 95%; background-color: #FFDAB9;"></div>
	popis PU	<div style="width: 70%; background-color: #ADD8E6;"></div>
	doménový model	<div style="width: 85%; background-color: #FFFF00;"></div>
	znalost požadavků	<div style="width: 70%; background-color: #8A2BE2;"></div>
4	aktéři	<div style="width: 100%; background-color: #90EE90;"></div>
	seznam PU	<div style="width: 100%; background-color: #FFDAB9;"></div>
	popis PU	<div style="width: 80%; background-color: #ADD8E6;"></div>
	doménový model	<div style="width: 90%; background-color: #FFFF00;"></div>
	znalost požadavků	<div style="width: 90%; background-color: #8A2BE2;"></div>
5	aktéři	<div style="width: 100%; background-color: #90EE90;"></div>
	seznam PU	<div style="width: 100%; background-color: #FFDAB9;"></div>
	popis PU	<div style="width: 90%; background-color: #ADD8E6;"></div>
	doménový model	<div style="width: 95%; background-color: #FFFF00;"></div>
	znalost požadavků	<div style="width: 100%; background-color: #8A2BE2;"></div>

Základní aspekty plánování

- plán je vždy nutný
 - termíny, harmonogram
 - pevné body
 - přiřazení zdrojů
- sledování plánu nutné vždy
 - kontrola postupu
 - reakce na změny
- způsoby plánování
 - prediktivní / adaptivní
 - rizika / priority / ROI...

Prediktivní plánování

- typické pro sekvenční postup
- WBS → PERT → Gantt
 - rozpis prací od začátku až do konce projektu
 - jasné milníky, jasné artefakty
- problém – velká míra nejistoty
 - neznalost odhadů v době, kdy jsou potřeba
 - požadavky se mohou měnit, takže i rozsah
- co s tím – zkušenosti, metriky, menší projekty

Adaptivní plánování

- detailně plánujeme jen to, na co máme data
- přesnější odhady a plán až po nějaké době
- plán na N+1 iteraci je zpřesněn z N iterace
- problém – náročnost a orientace
 - průběžné sledování a kvalitní management
 - zvenku může působit nesystematicky

- co s tím – globální pevné body plánu předem

Okamžiky pro plánování

- Na počátku projektu
 - hlavní cíle, hrubé odhady → pracnost, čas a zdroje = cena
 - globální řízení projektu
- Na začátku každé iterace
 - seznam aktivit a úkolů
 - odhadování pracnosti, jak času, tak zdrojů
 - řazení a vyřazení dle priorit
 - aktualizace plánu projektu možná
- V průběhu iterace, se neplánuje, pokud to jde

Jak vybrat aktivity a termíny

- do kdy má být co hotovo
- řízení riziky
 - vyhodnocení rizikových faktorů projektu
 - neznámá funkčnost, použitelnost
 - designová/architektonická rizika, obchodní, legislativní
 - začít částmi funkčnosti/designu s nejvyšší mírou rizika
- řízení prioritami klienta
 - určení pořadí výstupů je na zákazníkovi
 - množství omezeno časem
 - iterativní přístup umožňuje pružně reagovat na aktuální potřeby
 - začíná se částmi s největším významem pro zákazníka

Rizika

- přímá
 - projekt má velkou míru kontroly
- nepřímá
 - projekt má nízkou míru kontroly
- risk magnitude – klasifikace rizik
 - pravděpodobnost výskytu
 - dopad na project (na jeho zpoždění)

Priority v procesním plánování

- cílem je minimum nepoužívaných funkcí a vlastností systému
- přístup a techniky
 - vize produktu jako benchmark
 - plán na krátkou dobu
 - úrovně priorit – MoSCoW, RFC 2119
 - indikace důležitosti

Určování pracnosti

- klíčová součást plánování
- faktory jsou dopad, čas, rozsah, kvalita
- metriky
 - člověko-hodiny
 - obtížnost

- story points
- práce s časem – znát úvazek lidí a efektivitu procesu
 - ideální inženýrská hodina vs. režie
 - optimismus vs. pesimismus dle role

Dokumentace plánování

- Podklady
 - rozpad prací
 - zdroje a jejich alokace
- Grafické nástroje
 - PERT graf – návaznosti a kritická cesta
 - Ganttův diagram – čas a zdroje
- Dokumenty
 - Vize a rozsah produktu
 - Plán projektu
 - Plán pro řízení rizik
 - Plán iterace

Tvorba globálního plánu projektu

- výchozím bodem je vize, vytváří se při fázi zahájení projektu
- odhadnutí pracnosti (rámcové)- člověko-měsíce, hrubé WBS
- definice milníků – po stupních přesnosti a míře rizika, vodopád po činnostech
- rozdělit projekt na oddělené fáze – určení cílů a výsledků (1...N iterací)

Odhadování pro plán projektu

- určit pracnost → čas a zdroje
- problém závazek vs. znalosti
- vstupní informace
 - rámcový rozsah – vize – množství funkcí a charakter
 - očekávaný termín a ceny – smluvní podmínky
 - dostupné zdroje – lidi a technika – šéf vývojářů
 - metriky – historické údaje, průběh projektu
- postupy
 - zkušenosti
 - občas nástroje Wideband Delphi

Plánování iterace

- Cílem je vybrat množinu úkolů z product backlogu pro iteraci do sprint backlogu
- Východiska – vize, plán, požadavky (DSP, product backlog)

Planning Meeting

- naplánování miniprojektu
- Cíle
 - co má být výsledkem iterace
- Určení / výběr požadavků
 - podmnožina DSP odpovídající cíli, úroveň případů užití
 - dosud získané informace o požadavcích (zohledňujeme priority)
- Zpřesnění požadavků, odhad pracnosti
 - obsah vybraných požadavků + Done

- rozpad na úkoly (tasky)
- Definiton of done
 - **dodělej DoD**
- tasky
- Určení a commitment prací
 - co je nejdůležitější, co je reálné udělat → co teda bude uděláno
 - direktivní přístup, týmový přístup, planning game
- Vytvoření plánu
 - use case → task → sprint backlog

Obsah vybraných požadavků

- vyjasnění konkrétní funkčnosti pro realizaci
- pro plánování nutné odhady, pro ně details
- bereme z DSP, backlogu, Info od zákazníka, vize, architektura

Tasky

- konkrétní zadání pro členy týmu
- odvozeny z use case, user story
- forma je post-it, bugzilla
- konkrétní hodiny, max. 1 pracovní den

Odhadování pracnosti

- komplikací je, že neznáme a-priory dostatečně přesně
 - rozsah požadavků
 - míra přesnosti (nepřesnosti)
- Základem vždy forma Work Breakdown Structure + metriky
- Postupy
 - analyticky – WBS → pracnost, PERT → termíny
 - adaptivně – planning poker, wideband delphi
 - tým a zákazník
 - priority vs. odhady
 - commit
 - analogie – z předchozími projekty, historická data
 - odhadem

Sledování průběhu

- to je nutné
 - kvůli rozpoznávání blížícího se rizika
 - schopnost reagovat na změnu
- project cracking a oversight
 - odhadovaný vs. skutečně strávený čas
- metody
 - reporty – analytické nástroje
 - veřejně přístupný plán
 - komunikace - schůzky, XP role Tracker

Úpravy postupu

- výchozí zkušenosti
 - plán není nedotknutelný

- ani krátkodobé se vždy nepovedou
- uvnitř iterace
 - vyjímecné stavy a jejich řešení
 - je to nutný?
- mezi iteracemi
 - ideální řešení na změnu
 - viz. retrospektiva iterace

Určení výkonnosti týmu

- globální řízení iterativního vývoje
 - chceme dosáhnout vize
 - umíme odhadovat víceméně pouze iterace
- zákazník se ptá, kdy teda bude hotovo?
- team velocity
 - kolik uživatelsky užitečné funkčnosti dokáže tým dát za iteraci
 - tým (lidi) + požadavky (pracnost) + plán (čas) + změny (realita)
 - průměrná, přibližná hodnota => 3 iterace

Přednáška č.3 – fáze projektu Elaboration (projektování)

Přehled a cíle

- zahájený projekt (vize) → sběr a analýza požadavků → návrh architektury technického řešení → ověření návrhu → příprava na vývoj
- rozumně kompletní DSP – všechny klíčové, kritické požadavky
- architektura – základní rysy technického řešení
- infrastruktura – prostředí pro vývoj

Klíčové disciplíny a aktivity

- doladění vize
- sběr požadavků
- zpřesnění klíčových požadavků
- návrh architektury
- validace architektury
- výběr technologií a komponent
- ujasnění procesu
- příprava vývojového prostředí
- naplánování iterací pro realizaci

Charakteristika fáze

- důležitá pro nově zahajované a technicky složité projekty
- v pokračujících význam pro znovu ověření adekvátnosti řešení
- 1-2 hutné iterace
- význam klíčových rolí
 - analytik a architekt

Architektura systému

- Klíčové aspekty organizace
 - jak je systém členěn, jak něco dělá a proč to vlastně dělá

- technologie
- hrubé členění na subsystemy, závislosti a rozhraní mezi nimi
- aspekty prolínající celou implementací - konvence
 - jak systém navrhovat
 - vzory návrhu / implementace
 - mimofunkční aspekty

Kontext a principy

- Kontext (daný)
 - okolí systému → vazby a důvody pro rozhodnutí
 - stakeholderi → úhly pohledu → aspekty architektury
 - logická struktura, procesní pohled, varianty nasazení, datová integrace, bezpečnost, provoz a podpora, provozní infrastruktura, rozhraní...
- Principy architektury
 - volí si vývojáři
 - efektivita
 - srozumitelnost
 - jednotnost

Výběr technologie a omezení

- programovací jazyk, databáze, knihovny
- použití frameworků a hotových komponent
 - rozhodnutí udělej nebo kup
- faktory, které omezují nebo ovlivňují vývoj softwaru
 - smluvní podmínky
 - standardy (i lokální)
 - viz Enterprise architecture
 - kontext
 - marketing
 - technické znalosti

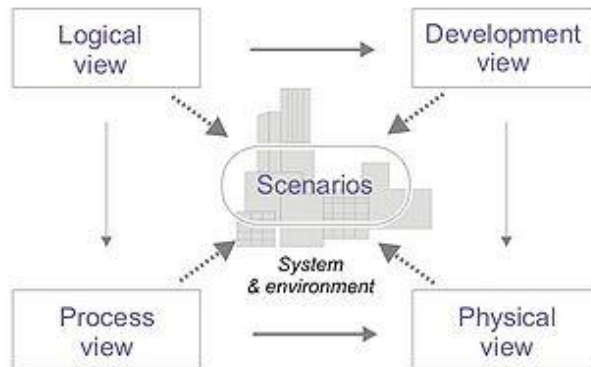
Zdůvodnění architektonických rozhodnutí

- proč je právě taková
- architektonicky důležitá funkčnost
- vztah k omezujícím podmínkám

Architektonické struktury – části architektury

- co je významné a proč
- RUP koncept, 4+1 pohled
- Obecný koncept – lokalita změn

4+1 pohled



Logický pohled

- motivací je, že monolitická aplikace je nepřehledná
- subsystém – skupina souvisejících prvků implementace tvořící funkční celek
 - funkčně soudržné
 - často vázaná na jednoho aktéra
- balík
 - agregace prvků implementace – jmenný prostor
 - snaha o přehlednost a srozumitelnost
 - oddělení veřejných a privátních prvků implementace
 - členěno pro získání
 - přehled o systému
 - rozdělení práce mezi členy týmu
 - balíky se hierarchicky vnořují
 - třídy balíku
 - funkčně příbuzné, v jedné vrstvě aplikace nebo kdekoliv

Fyzický pohled

- motivací je, že monolitická aplikace je nepraktická
- subsystém
 - samostatná správa – nasazení, údržba, přístup
- modul, komponenta, knihovna
 - celek vhodný pro samostatný vývoj, nasazení a údržbu
 - snaha o REUSE – vísenásobnou použitelnost
 - důležité kvalitativní charakteristiky
 - pravidla – information hiding, open-closed princip
 - metriky – složitost, fan in/out
 - nutnost znát a spravovat závislosti
- moduly mezi sebou komunikují pomocí rozhraní
- komponentový přístup
 - dotažení modularity, zapouzdření a rozhraní do konce
 - komponenta
 - rozhraní rozdělena na poskytovaná a vyžadovaná → vazby
 - explicitní specifikace rozhraní a vlastností
 - technologie
 - portlety, OSGi
 - Spring, PicoContainer
 - částečně CORBA, EJB

Od logické struktury k fyzické struktuře

- realizace logické struktury se provádí v konkrétních technických artefaktech
- tyto je potřeba vytvořit
 - jednotlivci, týmy, projekty
 - technologické aspekty
 - vývojářská infrastruktura
 - alokace prvků logické struktury do vývojových projektů
- Logický pohled – subsystemy, balíky → diagramy tříd
- Vývojový pohled – projekty, adresáře, sestavení
- Fyzický pohled – moduly, komponenty, knihovny

Konvence a politiky

- architectural policies
 - obecná pravidla pro návrh libovolné části aplikace
 - musí je dodržovat všichni vývojáři
- použité návrhové vzory
- správa paměti
- synchronizace, transakce
- defenzivní programování
- dokumentace kódu

Procesní pohled

- struktura paralelizace v implementaci
 - procesy, vlákna → jak synchronizovat
 - komunikace synchronní a asynchronní
 - propustnost, škálovatelnost a odolnost
 - podpora – OS, jazyk, knihovny
- vazba na strukturu nasazení (DS)
- alokace aktivit do modulů implementace
- synchronizace, předávání artefaktů

Dokumentace architektury

- referenční architektura
 - kostra aplikace
 - dokument
 - RUP Artifact – Software Architecture Document
- modely
 - komponenty, třídy, balíky, interakce, stavový model
 - ad-hoc – visio, tabule

Vazby mezi pohledy

- dáno souvislostmi mezi prvky systému
 - vazba artefaktů a logických celků
 - vazba artefaktů mezi sebou
 - aktivita nebo pasivita artefaktů za běhu
 - nasazení artefaktů na provozní infrastrukturu
- dáno postupem tvorby
 - prioritizace funkcí a omezení rizik ovlivňují postup vytváření
 - závislosti artefaktů definují postup skládání

- připravenost infrastruktury ovlivňuje nastavení

Kdy začít členit architekturu

- členění ovlivňuje plánování a postup prací
- standartní projekty
 - možno odhadnout nebo stanovit předem
- menší systémy
 - rozdělení na základě analýzy
- velké projekty (analýza trvá dlouho)
 - nahrubo předem → rozfázování prací včetně analýzy
 - vodítko → aktéři, znalost struktury fyzického systému, možnosti znovu použití, vývojové kapacity
 - přesně až během návrhu architektury

High-level architecture

- Základní hrubé členění celého systému
 - funkční subsystemy, vrstvy
 - vzájemné vazby
 - vazby na HW
 - vazby na okolní SW
 - důsledky na EFP
- Typické koncepty
 - vrstvy
 - filtry
 - služby
 - sdílená data

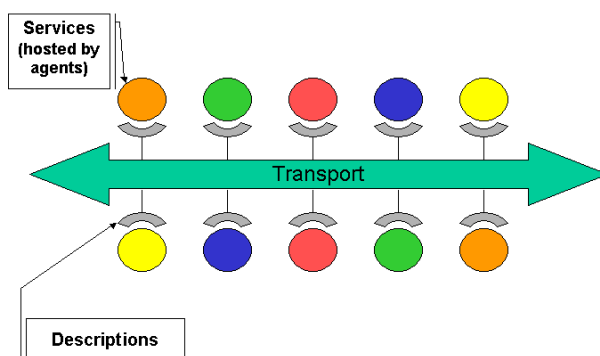
Vrstvená architektura

- delegování na podřízené
- komunikace se sousedy
- vrstvy například
 - prezentace
 - řízení
 - doména
 - business služby
 - technická infrastruktura
 - knihovny třídy
 - systémové třídy
- Klient-server
- Třívrstvé a vícevrstvé
 - oddělení prezentační, bussiness logiky a datové části

Další architektonické styly

- SOA – Service Oriented Architecture
- Pipes and filters – kolona
- Blackboard
- Broker
- Map-Reduce

Service Oriented Architecture



3 – vrstvý model (MVC)

- 3vrstvá architektura v malém
 - oddělení vrstev, které se nejčastěji mění
 - typicky UI
- Model-View-Controller
- Realizace pomocí tříd/objektů
 - hraniční – rozhraní
 - funkčnost přímo závislá na okolí
 - dialog s uživateli
 - komunikace s externími systémy
 - aktéři přistupují k systému pouze přes hraniční objekty
 - úlohy
 - prezentace a čtení informací
 - zapouzdření externích prvků
 - zpracování informací, jejich uchovávání
 - podle toho základní atributy a metody
 - objekty zřejmé z popisu rozhraní v případech užití
 - specifikace požadavků, znalost systému
 - odraz v prototypu
 - alespoň 1 hraniční objekt pro 1 aktéra
 - rozbor případů užití
 - vyznačení míst s rozhraním na systém
 - funkčnost závislá na podobě rozhraní
 - obvykle kombinace – konsolidace výsledků
 - řídicí – mechanismy
 - stavové přechody, mechanismy (business rules)
 - funkčnost, která se může změnit nezávisle na datech
 - některé nelze nebo není dobré svázat s hraničními / datovými třídami
 - vyčlenění do speciálních tříd
 - vlastnosti
 - obvykle jen pro daný případ použití
 - poměrně malé, těžiště v několika metodách
 - najdeme v rozbořech PU
 - transakční operace
 - izolace hraničních a datových objektů
 - zajištění komunikace mezi objekty
 - technika
 - pro začátek jeden PU → jeden objekt

- hledání typických chování
- více různých → výroba dalších řídicích
- typické chování není → řídicí objekt zbytečný
- datové – perzistence údajů
 - zapouzdřují informace, které se delší dobu uchovávají
 - atributy
 - uchovává informace, jednoduché i strukturované
 - metody
 - funkčnost svázaná s přístupem k informacím
 - životní cyklus objektu, přístup k datům, zpracování dat
 - jsou zřejmé z výsledků analýzy
 - často doménové objekty
 - data vstupující do systému
 - rozbor případů užití
 - vyznačení dat používaných při zpracování
 - přímo svázané operace
 - zapouzdření do objektů
 - separovat společné vlastnosti → dědičnost
 - realizace v SQL... až později
- navíc knihovny a systémové
 - realizace nízkourovňových úloh
 - komunikace, soubory, zobrazování
 - cílem reuse
 - znalost knihoven
 - namapování hraničních příp. datových tříd

Úvaha o strategiích alokace řízení

- hraniční vs. datové vs. řídicí objekty
 - každý může mít řídicí mechanismy
- cílem je minimalizace dopadu a lokality změn
 - uživatelské / systémové rozhraní
 - reprezentace a předávání informace
 - malá změna požadavků → lokální změna implementace
- aplikace podle typu řízení
 - výpočetní a řídicí systémy, dialogové, kombinované
 - vyvážené

Detailní návrh – design in a small

Zodpovědnosti a metody třídy

- cílem je interní funkčnost jednoznačně odvozená od vnější požadované
 - údržba
- rozbor případů použití
 - akce → zodpovědnost tříd
 - zodpovědnost → metoda
- jakmile potřebuji (zodpovědnost)
- jakmile je dost informací (metody)

Mechanismy spolupráce tříd

- mechanismy → více tříd pro realizaci jedné funkčnosti

- ideálně – zobecnitelné
- využití návrhových vzorů
 - listener, strategy,...
 - návrh postupu předávání zpráv
 - soulad s architektonickými principy a pravidly
- mechanismy rámují jednotlivé zodpovědnosti
 - rozbor scénářů případů užití
 - správně identifikovat spouštěcí událost

Vztah případů užití

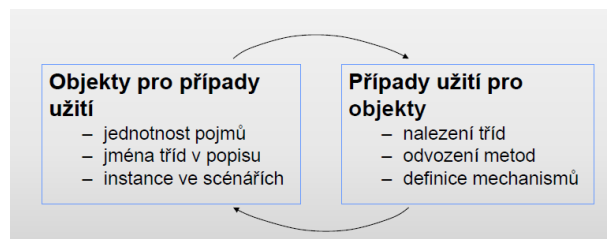
- scénář jednoho případu užití
 - popisuje konkrétní interakci instancí → mechanismy
 - generuje metody jejich tříd
 - model „Use Case Realization“
- soubor scénářů
 - vytváří celkový souhrn chování třídy
 - při vhodném výběru její úlohu → rozhraní
- výsledkem je tvorba nebo doplnění objektového modelu
 - třídy, metody, algoritmy, parametry

Stavové modely objektů

- přechody mezi stavy
 - na základě volání metod
 - podmíněné částmi interního stavu
- objekty řízené podněty
 - nezávislost na stavu – volání metod v libovolném pořadí
 - časté pro datové a servisní objekty
- objekty řízené stavem
 - stav určuje možná volání – pořadí metod tvoří protokol
 - přechody mezi stavy chráněné podmínkami
 - časté pro řídicí objekty

Souvislost analýzy struktury a dynamických aspektů

- iterativnost – základní vlastnost objektové analýzy a návrhu



Nutnost konsolidovat model

- z analýzy scénářů
 - pohled na jednu roli třídy → některé metody třídy
 - ze souboru scénářů všechny metody třídy potřebné pro realizaci jednotlivých případů použití
- typicky práce více lidí a rozložená v čase
- výsledkem je nekonzistence rozhraní
 - názvy metod, syntaxe volání

- duplikáty, podobnosti, rozpory v sémantice
- příčiny: různé scénáře, různí autoři

Úpravy rozhraní a vnitřku tříd

- Konsolidace syntaxe a sémantiky tříd
 - komplementarita metod (create → destroy)
 - znalost domény a implementace → další atributy a metody
 - snaha o minimalitu → reuse
 - viditelnost vlastností (private, public)
- Následné zpětné úpravy scénářů

Úpravy celkového chování třídy

- role třídy (zodpovědnosti, mechanismy) → sady metod, rozhraní
 - diagram tříd, komponent, kompozitů
- postup implementačního mechanismu → pořadí volání metod, protokol
 - obvykle stavové modely
- dohromady kontrakt třídy
 - důležitá součást specifikace návrhu

Úpravy vztahů a modelu

- úpravy objektového modelu
 - vyhledání opakujících se prvků → rodičovské třídy
 - vyjasnění vztahů (asociace, agregace, kompozice)
 - nové řídicí objekty, nové vztahy mezi třídami
- refaktoring
 - změna implementace bez změny funkčnosti
 - mění design za pochodu (v extrémním případě i architekturu)
 - podmíněno používáním automatizovaných testů

Rekapitulace návrhu a architektury

- známe skoro všechny detaily implementace
 - rozpracovaná funkčnost validuje architekturu
 - vazba na konkrétní technologie
 - vyřešená komunikace s okolním prostředím
 - struktura aplikace a rozhraní mezi částmi
 - používané konvence a návrhové vzory
- dál naprogramovat co zbývá a integrační testy

Detaily v UML modelu užití

Detailní analýza požadavků

- fáze projektování (rozpracování)
- známe
 - zadání a cíl projektu
 - přehled důležitých funkcí
 - možné zdroje problémů
- dále
 - doplnit detaily požadavků
 - najít business mechanismy

- vymyslet třídy pro jejich realizaci

Určení detailů případu užití

- najít všechny scénáře
 - vyjasnění nejednoznačností v zadání
 - detaily normálního průběhu
 - všechny alternativy
- určení výsledného stavu systému
- očekávání změn
 - primární a sekundární PU
 - společná funkčnost → vkládané a zobecněné PU
 - změny v aktérech, sekundární aktéři

Vztahy mezi PU

- Zdroje vztahů
 - z povahy věci, snaha o zobecnění
- zahrnutí jiného PU
 - rozšiřuje funkčnost, reuse vkládaného
 - kam vložit – uvést u vkládajícího
- rozšíření PU o jiný
 - původní průběh nezměněn
 - přidány nestandardní průběhy
 - kam vložit – uvést u vkládaného
- generalizace základního průběhu
 - specializované PU doplňují detaily

Výsledný model případů užití

- úplnost
 - všichni aktéři
 - popis
 - 20% případů na 99%
 - 60% na 60%
 - 20% na 1%
 - pozor na samozřejmosti a skryté požadavky
 - včetně technických PU a reuse scénářů
- přehlednost a srozumitelnost
 - rozložení diagramu, struktura modelu
 - hodně PU → package, více diagramů
- textové popisy
 - srozumitelné pro zákazníka
 - forma dokumentu – schvalování a kontrakt

Systemové sekvenční diagramy

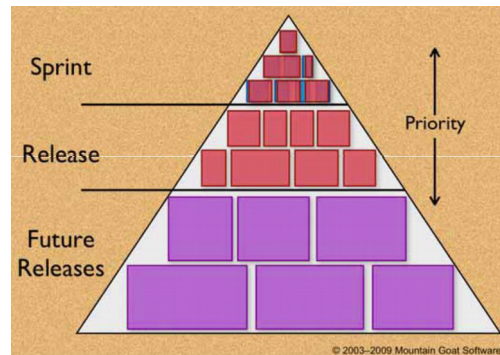
- shrnují komunikaci aktér – systém

Forma user stories

- uchování
 - papír, bugzilla, excel, xplanner
- důležitá je flexibilita práce
- doplňující položky
 - prioritita (MoSCoW)
 - důležitost (1...100)
 - pracnost (story points)

Dokumentace – způsob

- stejný – product backlog



Vlastnosti systému

- je třeba vědět nejen co, ale taky jak dobře
- mimofunkční požadavky
 - doby odezvy, spolehlivost, objemy
 - nutný doplněk požadavků na funkce
 - dopad na architekturu, implementaci (někdy významný)

FURPS+

- model třídění vlastností
 - funkcionality, použitelnost, spolehlivost, výkon, podporovatelnost + omezení
- omezující podmínky
 - normy a zákony
 - obchodní pravidla
 - implementační omezení (technologie, rozhraní)
 - fyzické charakteristiky

Business rules

- popis faktů o fungování business logiky (know-how)
 - volně přiřazená k datům, procesu
 - často platná globálně v celém systému
- strukturální
 - jak se co z čeho tvoří
 - pravidla integrity systému
- chování
 - pre a post conditions

Procesní modely

- popis scénáře funkčnosti při složitém rozhodování
- když je text nepřehledný
- nadřazený proces dělený do funkčností (PU)
- notace
 - UML diagram aktivit
 - DFD

Chování aplikace

- typicky chování UI
 - správa dat, průvodci, přihlašování
 - náhradu modelu UI
- Prototyp uživatelského rozhraní
 - papír → PPT → Demo
 - reprezentace vnějšího rozhraní produktu nebo jeho části, v abstraktní či konkrétní podobě
 - podklad pro určování funkcí a ovládání aplikace
 - diskuze nad prototypem → korekce neshod v porozumění
 - diskuze k prototypování
 - jednoduchá testovací data
 - abstraktní podoba UI
 - řízený konečným automatem
 - kam s ním
 - vyhodit vs. uchovat
 - hraniční třídy pro objektový návrh
- stavový model
 - stav – obrazovka , krok průvodce, ...

Formální metody

- matematický model chování aplikace
- dokazatelná
 - správnost, úplnost, bez rozporů
- specializované notace
 - Z,VDM, B-method
- nástroje...

Modelování datové části

Použití doménového modelu

- komunikace
 - dorozumění s klientem
- objektová analýza a návrh
 - s těmito třídami můžeme najisto počítat
 - analýza přidá zodpovědnosti, detaily vlastností a chování
 - návrh je transformuje a přidá implementační třídy
- datové modelování
 - doménový model → logický datový model

Datový model

- logický (konceptuální), fyzický

- obvykle jeden z prvních nezávisle na OO technikách

Vývoj datových entit

- entita typicky prochází vývojem
 - životní cyklus
- model → stavový diagram
 - vazba na doménový / datový model

CRUD(L) matice

- cílem je vědět, kdo nebo co manipuluje s jakými údaji
- Create-Read-Update- Delete-List
- Úroveň detailu
 - analytická – uživatel, případ užití vs. informace
 - návrhová – třída, proces, funkce vs. tabulka

Přednáška 4. Fáze Construction (konstrukce)

Přehled a cíle

- známý cíl a technická architektura → dovyrobit produkt a ujasnit při tom zbylé / příslušné požadavky
- dosáhnout nasaditelných verzí co nejrychleji v co nejlepší kvalitě
- připravit nasazení do provozu

Klíčové disciplíny a aktivity

- zpřesnění požadavků
- úpravy návrhu
- vývoj a testování modulů
- ověření vydávaných verzí dle vize
- řízení zdrojů, dohled nad procesem

Charakteristiky fáze

- důraz na efektivitu a kvalitu
- obvykle 2 a více iterací
 - fáze krátká v integračních projektech
- klíčové role
 - tester, vývojář
 - technický manager / chase kontrol board
 - podpůrné

Výchozí problémy a motivace

- při vývoji produktu ve více verzích a nebo lidech je nutné zamezit zmatkům při
 - realizaci jednotlivých uživatelských požadavků
 - implementaci změnových požadavků přidávaných za pochodu
 - oprava chyb na různých verzích
 - provádění změn na jednom objektu (dokumentu, tabulce)
 - vytváření a označování spustitelné verze
 - zjišťování aktuálního stavu vývoje, nasazené verze

Konfigurační management

- proces identifikace a definice prvků systému
- řízení změn těchto prvků během životního cyklu
- zaznamenání a oznamování stavu prvků a změn
- ověřování úplnosti a správnosti prvků
- IEEE-Std-729-1983
- SCM – software configuration management

Prvek konfigurace

- konstituující složka systému
 - zdrojový soubor, model, knihovna, dokument, skript, spustitelný soubor, testovací data
- ve správě SCM
 - ví se o jeho existenci, vlastníkově, změnách, umístění v produktu
 - atomický z hlediska identifikace a změn
 - jednoznačně identifikovatelný
 - typ prvku
 - označení projektu
 - název prvku
 - identifikátor verze

Konfigurace

- SW konfigurace
 - sestava prvků konfigurace reprezentující určitou podobu daného SW systému
 - musí tam být vše, co je potřebné k jednoznačnému opakovatelnému vytvoření příslušné verze produktu
 - překladače, build skripty, inicializační data, dokumentace
 - může být jednoznačně identifikovatelná
- Konzistentní konfigurace
 - její prvky jsou navzájem bezrozporné
 - zdrojové soubory jdou přeložit, knihovny přilinkovat
 - těsná souvislost SCM a QA

Popis konfigurace

- struktura produktu jako množiny prvků a jejich vzájemných stavů
- prvek → výsledek SW procesu
 - různá granularita a reprezentace
- vztahy
 - celek-část, master-dependent → určují strukturu a závislosti
 - zdrojový-odvozený → určují způsob produkce – build produktu

Úlohy SCM

- určení a správa konfigurace – CFG identification and control
- zjišťování stavu systému – status accounting
- správa sestavení a koordinace prací – repase management

Aktivity SCM v cyklu vývoje

- správa změn
- identifikace a správa verzí
- sestavení

Správa změn

- jak zvládat množství požadavků na úpravy produktu? jak poznat, že jsou vyřešeny? jak dohledat, co bylo změněno?
- nutný striktní postup akcí
 - vyřešení prioritních, udržení konzistence a stability
 - informovanost o změnách, prevence duplikování práce
- význam ve všech fázích ŽC
 - evidence požadavků během jejich sběru
 - přiřazení příslušné práce během vývoje
 - hlášení a opravy nalezených chyb při testování
 - úpravy produktu během provozu a údržby

Ticket, požadavek na změnu

- hlášení problému (bug report), feature request – popis nalezeného nebo požadovaného vylepšení
 - někdy obecně ticket
 - strukturovaný dokument, obvykle v ALM nástroji
 - zdrojem jsou QA aktivity, uživatel, marketing
- požadavek na změnu (chase request) – popisuje změny, které se mají provést na prvku (prvcích) konfigurace
- často spojován do formy jednoho dokumentu

Hlášení projektu – detaily

- při vytvoření
 - nutné náležitosti (id, autor, datum)
 - co nejpřesnější popis
 - jak chyba vznikla
 - jak jí reprodukovat
 - screenshot, vzorek dat
 - závažnost, priorita
 - konfigurace daného SW a systému (OS, knihovny,...)
 - identifikace verzí
- při vyhodnocení
 - závažnost, priorita
 - odhad pracnosti (přímé – kód, návazné – doc)
 - komponenta, verze
 - závislosti
 - zodpovědný vývojář
- po uzavření
 - shrnutí a zdůvodnění
 - skutečná pracnost
 - výsledná revize souborů / aplikace

Akce při zpracování požadavku

- vytvoření a přijetí
 - přidělení ID
- vyhodnocení
 - možná řešení, odhady pracnosti a dopady
 - doplnění popisů, metadat
- rozhodnutí

- způsob vyřízení (vyřešit – odmítnout – duplikát – odložit)
- závažnost, prioritita
- naplánování a přidělení
- zpracování
 - příslušné požadavky na změny
 - komentáře, diskuze, související sady změn (commity)
- uzavření
 - build – ověření konzistence a verzování, merge
 - informování zadavatele hlášení a další zájemci

Případy nouze

- kdy porušit pravidla
 - marginální oprava těsně před repase
 - vyřešení problému u zákazníka
- pravidla pro porušování pravidel
 - jasný přínos výjimky
 - nekamuflovat
 - každý musí vidět, že nebyl dodržen standardní proces
 - zdůvodnění, proč se tak stalo
 - zpětně zdokumentovat provedené akce
 - opakované porušení musí vést k úpravě procesu

Change control board – CCB

- skupina členů projektu, která má zodpovědnost za změnové řízení
 - vyhodnocování a schvalování hlášení problémů
 - rozhodování o požadavcích na změny
 - může významně měnit chod a podobu projektu
 - koordinace s vedením projektu
 - sledování hlášení a požadavků při zpracování
- složení
 - jedinec – vývojář, QA
 - tým - technické i manažerské role
 - vhodné, pokud má změna velký dopad
 - znalost účelu produktu

Souvislosti správy změn

- správa změn není izolovaná aktivita
- správa změn a řízení projektu
 - kritéria dodávky (rebase management)
 - časový termín
 - funkčnost
 - kvalita dodávky
 - jak určit termín podle kvality?
 - konverguje nám iterace?
- správa změn a verzování
 - cílem je trasovatelnost
 - vyhodnocení požadavků, jaké verze se týká
 - uživatelská verze
 - interní verze
 - uzavření požadavků

- do BT výslednou verzi
 - do správy verzí ID vyřešeného požadavku
 - vazba ticket-commit klíčová
- správa změn a požadavky
 - požadavek = feature request
 - workflow
 - vize
 - požadavky → DSP
 - feature request → bugtracker
 - (testy)
 - bug report a nebo re-open feature request
 - flexibilita, souvislost s plánováním
- správa změn a údržba
 - provoz produktu (uživatelská podpora)
 - vs. aktivní vývoj
 - nutná o to větší pečlivost při úpravách
 - hlášení problému
 - oprava
 - garance, servisní smlouva → více práce
 - požadavek na vylepšení
 - ihned → více práce
 - naplánovat do přírůstku

Systemy pro správu změn

- BT – bug trackery
 - flyspray, mantis, bugzilla
 - jednoduché, snadná instalace, informace emailem, webové rozhraní
- ALM – application lifecycle management
 - redmine, RTC
 - velké projekty, konfigurovatelné

Správa verzí

- udržení přehledu o podobách prvků konfigurace a konfigurací
- verze – jedna konkrétní podoba prvku nebo konfigurace
 - podoba → druh verzování
 - co je verzováno → granularita
 - jak je verze určena → typ
- nástroje jsou úložiště a verzovací systém (VCS)

Určení konkrétní verze prvku

- základní druhy verzí
 - historická podoba → revize (Word 6.0)
 - alternativní podoba → varianta (Word pro Mac)
- verzování podle stavu
 - identifikují se pouze podoby prvků
 - výsledná verze vznikne vhodným výběrem
- verzování podle změn
 - identifikují se změny prvků – changesety
 - výsledná verze vznikne aplikací změn

Granularita verzování

- verzování komponent
 - jednotlivé prvky samostatně
 - konfigurace nemá verzi
- úplné verzování
 - verzi má celá (sub)konfigurace
 - indukuje verze prvků
- produktové (uniformní) verzování
 - struktura (sub)konfigurace a systém verzování nezávislé
 - výběr verze indukuje prvky konfigurace

Identifikace konkrétní verze

- extenzionální
 - každá verze má ID
 - jednoduchá implementace
 - problémy při větším počtu verzí
 - naprostá většina CVS
- intenzionální
 - verze popsána souborem atributů
 - nutné pro větší prostory verzí
 - potřeba vhodných nástrojů

Nástroje pro verzování

- úložiště (DB projektu, repozitář)
 - sdílený datový prostor, kde jsou všechny prvky konfigurace
 - centrální místo
 - všechny prvky ve všech verzích
- nutný řízený přístup (konzistence)
- version kontrol systém (VCS)
 - sada nástrojů pro práci s úložištěm

Práce s úložištěm

- základní operace
 - vytvoření
 - inicializace
 - check out
 - Check in (commit)
 - branch + merge
 - tag
 - zjišťování stavu
- přístup zamykání při CI/CO
 - read-only pro všechny
 - pesimistický
 - read-write jen pro pověřeného
 - optimistický
 - read-write pro všechny ,řešení konfliktů

Centrální vs. distribuovaný vývoj

- je tým geograficky distribuovaný? v čase distribuovaný? potřebují členové pracovat online a synchronizovat? musí existovat jedna sdílená pravda? chci snadný fork projektu?
- možnosti
 - centrální úložiště
 - privátní větve
 - distribuovaný verzovací systém
- typické přístupy
 - CVS, Subversion vs. GIT, RTC
 - unikátní ID commitu, HEAD vs TIP
- vede na různé způsoby práce
- centralizovaný
 - update, commit
 - online přístup, zodpovědnost
- distribuovaný
 - fetch, update, commit, cleanup, push
 - znalosti a disciplína, rozlišení zdrojového úložiště
 - poskytuje variabilitu procesu

Pracovní prostor

- workspace – soukromý datový prostor, v němž je možno provádět změny prvků konfigurace, aniž by byla změněna oficiální verze
 - vývojářský soukromý
 - integrační sdílený
- zpřístupnění úložiště pro práci
 - clone
 - checkout

Pravidla na codeline

- policy – pravidla práce
 - klíčová součást
 - odlišení různých codelines
 - typ prací
 - očekávaná kvalita kódu
 - pravidla/akce před commitem, po checkout...
 - jak a kdy či co, merge, branch
 - přístup pro osoby a skupiny
 - kam exportovat změny, odkud importovat
 - doba platnosti či podmínky odstavení

Delta, diff

- množina změn prvků konfigurace mezi dvěma po sobě následujícími verzemi
 - přidání sekce kontext produktu do DSP
- v některých systémech jednoznačně identifikovatelná
- changeset – delta + důvod
- diff a patch
 - rozdíl mezi verzemi
 - aplikace rozdílu na verzi

Tag, baseline

- tag, label
 - označení konfigurace symbolickým jménem
- baseline
 - konzistentní konfigurace tvořící stabilní základ pro produkční verzi nebo další vývoj
 - stabilní je vytvořená, otestovaná, schválená managementem
 - změny prvků baseline jen podle schváleného postupu
- při problémech návrat k tag, baseline

Význačné baseline

- interní release - iterace
- milníky projektu
 - jednotlivé fáze životního cyklu
 - vodopád, spirála, iterativní
 - alfa verze – všechny featury, interní testování
 - beta verze – testování u vybraných zákazníků
- finální repase produktu
 - verze dodaná zákazníkovi na trh

Paralelní práce na stejné konfiguraci

- důvodem jsou velké úpravy, repase, spekulativní vývoj, varianty
- cílem je vzájemná izolace paralelních prací tak, aby ukládané změny během nich neovlivnily ostatní
- cena za izolaci - řešení konfliktů

Větvení a spojování

- Kmen (trunk, master) – hlavní vývojová linie
- Větev (branch) – paralelní vývojová linie
 - operace vytvoření větve (branch-off, split)
- Spojení (merge) – sloučení změn na větvi do kmene
 - slučuje se delta od branch-off nebo posledního merge
 - řešení konfliktů – automatizace dobrá, ne vždy možná)
 - 2-way 3-way merge

Graf verzí

- historie vývoje konfigurace
- zobrazení vazeb mezi verzemi
 - uzly – verze, hrany – vazby verzí
 - grafická podoba codeline

Postupy při verzování

- check-out výchozí verze
- vývoj (kmen, větev)
- lokální testy a opravy
- check-in – merge nově větve
- integrační testy a opravy
- tag (příp. nová baseline)

Nástroje pro verzování

- ruční verzování
- základní – správa verzí souborů
 - centrální úložiště
 - ukládání všech verzí v zapouzdřené formě
 - rcs, cvs, subversion
- distribuované
 - více úložišť, synchronizace
 - flexibilnější postupy
 - SVK, GIT, Mercurial
- Pokročilé – integrace do CASE
 - kombinace ext a int verzování
 - automatická podpora pro ci/co prvků z repa do nástrojů
 - Adele, ClearCase

Co má nástroj umět

- operace s úložištěm
 - ci, co, add, rename, move, import, export
 - stav prvků, diff, historie změn
- verzování
 - ci, co, data, revize
 - branch, merge, značkování
- podpora týmu a procesu
 - vzdálený přístup
 - konfigurovatelné zamykání a přístupová práva
 - automatické oznamování
 - spouštění skriptů při operacích
 - integrace do IDE, řádkové a webové rozhraní

RCS – Revision Control System

- správa verzí pro jednotlivé textové soubory
 - UNIX, Widle, DOS
- ukládá do foo.c v souboru
 - historie změn v textu souboru
 - autor, datum čas
 - textový popis změny zadaný uživatelem
 - další informace
- používá diff(l) pro úsporu místa
 - poslední revize uložena celá
 - předchozí delta vygenerované
- funkce
 - zamykání souborů
 - symbolická jména revizí
 - možnost větvení a spojování změn z větví do kmene
- složky (utility) z příkazové řádky
 - ci, co, rcs, rlog, rcsdiff, rcsmerge

CVS – Concurrent Versioning System

- práce s celými konfiguracemi najednou

- sdílené úložiště + soukromé workspace
 - lokální nebo vzdálené úložiště
 - zjišťování stavu prvků, rozdílů oproti repu
 - možnost definice obsahu a struktury konfigurace
 - trigger
 - vše co umí rcs
- free software
 - původně nadstavba rcs
 - příkazová řádka, grafické nadstavby
 - integrace do mnoha IDE a CASE nástrojů

SVN – Subversion

- následník CVS
- bez omezení předchůdce – přejmenování, verzování adresářů, atomický commit, http přístup
- nové možnosti – binární diff, meta-data, DAV – abstraktní síťová vrstva, čisté API
- způsob práce a příkazy podobné CVS
- identifikace verzí
 - číslují commity
 - není koncept značek jako CVS
- obecná operace copy
 - kopíruje jednu část úložiště na jinou část
 - význam kopie dle potřeby
 - značky
 - vytvoření větve
- doporučená struktura úložiště
 - oddělená adresáře pro kmen
 - větve a značky

GIT

- distribuovaný verzovací nástroj
- repository
 - master, lokální
 - origin
- operace
 - clone, push, pull
 - rebase, revert, cherry-pick, stash

GitHUb

- nadstavba GIT
- hosting úložišť, jednoduchý project management
- social coding
 - snadné řízení příspěvků – pull request

Řízení sestavení

- aktivity provádějící transformaci zdrojových prvků na odvozené
 - zejména sestaven celého produktu
- vytvoření systematického a automatizovaného postupu
- pojmy
 - build – proces a vytvoření částečné nebo úplné podoby aplikace
 - zdrojové a odvozené prvky konfigurace

Postup při vytváření sestavení

- build process
 - míra formálnosti a preciznosti
- kroky
 - příprava
 - check-out
 - preprocessing, překlad, linkování
 - nasazení
 - spuštění
 - testování
 - značkování
 - informování

Součásti prostředí pro sestavení

- pravidla (nemenit)
 - vývojová linie
 - součást a vlastnosti sestavení
- skripty
 - check-out, značkování, check-in
 - preprocessing, překlad, linkování
 - informování vývojářů, tvorba statistik
 - vytvoření distribuční podoby (packaging)
- vyhrazený stroj (workspace)
 - build machine
 - zejména pro integrační a release sestavení

Typy sestavení

- co je použito pro sestavení
 - čas překladu vs. jistota správnosti
 - čistý
 - úplný
 - přírůstkový (inkrementální) build
- účel sestavení
 - lokální (neoficiální) komponenty povoleny
 - soukromý
 - integrační
 - release build

Vzory pro sestavení

- základní postupy
 - soukromé sestavení + sdílení součástí
 - integrační sestavení
 - release build
- podpůrné aktivity
 - kusovník a zapouzdřená identifikace
 - archivace prostředí
 - balení a distribuce
- obecný cíl
 - odchytit co nejdřív okamžik, kdy se to rozbilo

- QA-related postupy
 - zkouška těsnosti (smoke test)
 - regresní testy (regression test)
- typické workflow
 - daily build a smoke test
 - continuous integration
- obecný cíl
 - opakovatelná záruka dostatečné kvality

Soukromé sestavení

- spolehlivě ověřit, že jde produkt sestavit
- sestavit produkt v soukromém prostoru

Kusovník, archivace prostředí

- kusovník
 - kompletní seznam prvků sestavení
 - reprodukovat lze kdykoliv, kdekoliv
 - zejména při distribuovaném nebo jinak složitém buildu
 - samoidentifikační konfigurace pomůže
 - znalost verzí bez přístupu k verzovacímu systému
 - strojírnost, automatizace
- archivace prostředí
 - správa verzí objektů, které nejsou v úložišti
 - klíčové pro dlouho žijící SW

Release build

- význačné integrační sestavení – dodáno zákazníkovi
- náležitosti
 - revize/verze použité pro sestavení
 - datum vytvoření
 - identifikátor sestavení
 - další metadata
 - zodpovědná osoba
 - zdrojová značka konfigurace
 - jakými testy prošlo a výsledky
 - cesta k logům z překladu
 - marketingová verze

Základní struktury QA

- quality assurance
 - dobrý a dodržovaný proces
 - kontroly
 - statické ověřování
 - testování
 - metriky
- verifikace (bezchybný) a validace (správný) produkt

Testování

- ověření správné funkčnosti implementace
- úrovně testování
 - jednotkové → integrační → funkční → systémové
 - zátěžové, výkonnostní, bezpečnostní, instalační,
 - interní, akceptační
- techniky pro testování
 - white-box, black-box
 - výběr dat (hraniční podmínky, fuzzy testing...)
- automatizace

Statická kontrola kódu

- formální správnost, dodržování metriky, pravidel
- nástroje
 - překladač a jeho hlášení
 - C : lint
 - Java: pmd, findbugs, checkstyle
- postupy
 - programming by contract
 - review, párové programování
 - automatický build – výběr pravidel, průběžné úpravy

Smoke test (zkouška těsnosti)

- ověřit, že sestavení vytvořilo funkční produkt
 - samotný překlad a linkování nezaručí
 - kompletní testování trvá dlouho
- vytvoření testů ověřujících základní funkčnosti, bez nároku na kompletní otestování
 - odchytí nejkřiklavější chyby, odpustí drobnosti
 - automatizace – spuštění, vyhodnocení
 - spuštění při každém buildu
 - podle toho náročnost, rozsah
 - může být založena na testech modulů
 - přidání nových fcí a vlastností → nové testy těsnosti

Regresní testy

- zajištění, aby nové funkce a vlastnosti nezhoršili kvalitu kódu
 - prevence stárnutí kódu
 - vymyslet mechanismus, jak na vhodné testy
- ověřit build produktu pomocí testů, kterými již dříve prošel
 - indikátor existence systémových problémů
 - testování změn v integračních aspektech
 - spuštění při integračním sestavení, před velkými změnami
- dobrý zdroj testů – chyby objevené v QA, při validaci, zákazníkem
 - produkt selhal → napíšu test, který to dokáže → přidám do sady regresních

Měření kvality produktu

- pokrytí testy – kódu a požadavků
- charakteristiky defektů – hustota a výskyt
- kvalita zdrojového kódu
- junit → cobertura
 - zasévání chyb

Continuous integration

- dotažení do dokonalosti nebo extrému
- 1x denně → neustále
- klíčem je automatizace
 - co, ci, sestavení, testování oznamování
 - robot na spouštění

Nástroje pro podporu sestavení

- skriptovací jazyky
 - shell, perl, python, php
- buildovací nástroje
 - make, ant, maven, gradle
- verifikace
 - xUnit, junit, Selenium, testovací roboti
- proces sestavení
 - hudson/jenkins, cruisecontrol

Make – příklad závislostí mezi prvky

- build projektu na základě popisu závislostí typu zdrojový-odvozený
- pojmy
 - pravidlo
 - cíl
 - závislost
 - příkaz
- makefile
 - definice pravidel
 - příkazy pro překlad

Maven

- deklarativní build
- popis struktury projektu
 - project object model
 - komponenty, závislosti
 - paginy
- build automaticky
 - pro většinu jazyků a typů projektů předdefinované tasky - postup
 - získávání artefaktů → závislostí – úložiště centrální a lokální

Hudson/Jenkins/Travis

- automatický build a průběžná integrace
 - vyhrazený stroj
 - webovka
- spuštění buildu
 - post-commit hook, VCS pooling
 - upstream project
- konfigurace buildu
 - závisí na vlastním build nástroji
- informace
 - dashboard projektu, statistiky
 - drilldown (důvody neúspěchu, logy, konzole)

Měření v software

- k dosažení požadované kvality potřebujeme
- kvantitativní ukazatele
 - pomáhají najít slabiny → zlepšení
 - dávají přehled a kontrolu
 - kalibrují odhady
- Výhody
 - přesnost a dokazatelnost
 - možnost statistik a vizuální prezentace
- potřeba Organizational focus – záměr zlepšení kvality
 - vede k potřebě mít informace

Metriky

- metrika je měřitelná charakteristika nějaké entity
- měřený objekt – entita
 - produkt, proces
- rozměr
 - relativní nebo absolutní
- hodnoty
 - spojité, diskrétní, interval, výčet → operace
 - očekávané (min, max, avg), dosažené (dtto, trend)

Teorie měření

- metrika
 - získána na základě dat (primitivních metrik)
 - uvedena na stupnici – nominální, ordinální, intervalové, poměrné, proporcionální, poměrné, procentuální, míra
- platnost (validnost) měření
 - korelace, závislé a nezávislé veličiny
- spolehlivost měření
 - průměr, medián, odchylka
- chyby měření
 - systematické, nahodilé

Metriky SW produktu

- složitost, přehlednost
 - McCabe cyclomatic complexity
 - fan-in, fan-out → stabilita
 - weighted method per class
 - lack of cohesion
- velikost
 - počet UC, funkčních bodů
 - LOC – Lines Of Code
- spolehlivost
 - $MTBF = MTTF + MTTR$
 - dostupnost [%] = $(MTTF/MTBF) * 100$
- kvalita (nepřímé metriky)
 - pokrytí testy – kódy, požadavky
 - charakteristiky defektů – hustota, výskyt
 - kvalita zdrojového kódu
- nástroje
 - PMD, FindBugs, Cobertura/Emma
 - IDE nástroje

Projektové a procesní metriky

- postup
 - pracnost
 - project velocity / burndown
 - burnup – sledování dostupných zdrojů
 - jitter – change requesty a jejich rozpracování, staff turnover, změny postupu / plánu
- kvalita
 - breakage – průměrná váha změny (LOC, CR)
 - pracnost celkem, přepočtená na CR
 - defekt discovery rate, defekt removal (zpracování, trendy)
 - průměrná doba opravy

Jak měřit

- ukazatele sami o sobě k ničemu → měření jako proces
 - odhalování příčin, trendy, statistiky
- přístup k měření – ve firmě
 - proč měřit
 - jak s daty pracovat
- přístup k měření – pro projekt
 - co měřit
 - jak měřit
 - jak se zachovat, když to vyjde špatně
- techniky
 - GQM
 - zasévání chyb

Nástroje pro měření

- spreadsheet, kalendář
- bug tracker
- statsvn
- junit a cobertura
- databáze

Plánování a řízení měření

- organizační focus jako východisko
 - management musí chtít evidence-based process
- plán měření
 - proč měřit, co měřit, jak měřit, jak s daty pracovat, jaké akce provádět s výsledky
 - definice metrik, jejich význam a zpracování
 - způsob získání dat – příprava nástrojů
- sledování projektu a produktu
 - automatické získávání a vyhodnocování
 - sledování
 - korektivní akce
- komplexní přístup – program měření
- agile přístup – měření za pochodu

Přednáška 5 – fáze předání produktu (Transition)

Přehled a cíle

- hotová beta verze → dát produkt k dispozici uživatelům
 - triviální až extrémně složitá fáze
- fine tuning a dodávka
- na konci fáze předání by měli být splněny milník LCO → význam vize
- uzávěrka projektu

Přednáška 5 – fáze předání produktu (Transition)

Přehled a cíle

- hotová beta verze → dát produkt k dispozici uživatelům
 - triviální až extrémně složitá fáze
- fine tuning a dodávka
- na konci fáze předání by měli být splněny milník LCO → význam vize
- uzávěrka projektu

Charakteristiky fáze

- počet a povaha iterací různé
 - online služba
 - malý krabicový produkt
 - nová verze systému řízení rozvodové sítě
- možný současný / následný náběh nového cyklu
- legislativní a technické důsledky
 - záruka, servisní smlouva, podpora a údržba

Průběžné nasazování

- varianta celého vývojového procesu, nikoliv fáze předání
- kontext = webové systémy a služby
 - rychlé zpřístupnění nových funkcí a oprav koncovým uživatelům
- postupy a nástroje
 - průběžná integrace, plně automatizované testy
 - paging environment, automatizace nasazení
 - proces schvalování
 - DevOps

Příprava release

- nutné vs. obvyklé aktivity
 - field testing a korekce
 - konfigurace, použitelnost
 - pokud funkčnost pak je problém
 - konverze dat
 - příprava překlopení (cutover)
 - školení uživatelů
- pečlivost a příprava

Akceptační řízení

- formální odsouhlasení dodávky zástupcem uživatele
 - produkt a doprovodné materiály splňují požadavky
 - bylo dosaženo cílů vytyčených v plánu projektu
- následuje po úspěšných uživatelských testech produktu
- role
 - zástupce zákazníka
 - project manager, vedoucí týmů / oddělení
- podklady
 - iterativ assessment
 - výsledky testů a dalších QA aktivit
 - prohlášení o ukončení přechodu na nový systém
- výsledek
 - akceptováno vs. s výhradami vs. neakceptováno
 - zákazník přebírá produkt do vlastnictví

Uzávěrka projektu

- formální ukončení projektu, shrnutí zkušeností
 - zcela poslední iterace projektu
- interní audit konfigurace
 - kontrola úplnosti baseline (release) → fyzický audit
 - kontrola správnosti baseline → funkční audit
 - soupis chybějících prvků, neotestovaných funkcí, nefunkčních částí, neuzavřených change registů → opravy
- post-mortem review
 - pouze na základě analýzy našich nedostatků se můžeme naučit dělat lépe
 - faktory – dokumentace, komunikace, věcnost
 - možný postup – dotazník → data projektu → meeting a debata v týmu → data a příčiny → reporty

Nejdelší období životního cyklu SW

- produkce – udržení užitečnosti a efektivity
- podpora – support

Provoz a údržba produktu

- operations a support
 - zajištění provozu produktu, monitoring sítě, zálohování
 - pomoc uživatelům, analýza problémů, reporty chyb a rozšíření, nasazení oprav
- spolehlivost a dostupnost
 - důsledky na reálný čas pro odstávku

Vyřazení produktu

- odebrání z provozu a nahrazení novým
- důvodem je neúdržba, zastaralost, nepotřeba, náhrada novým
- aktivity
 - analýza vazeb na ostatní systémy
 - určení strategie a postupu vyřazení + nahrazení
 - testy, aktualizace dokumentace
 - migrace dat, uživatelů
 - archivace dat, kódu, dokumentace
 - vypnutí přístupu a konektorů
 - odstranění systému

Disciplíny pro enterprise kontext

- enterprise administrative
 - jak organizace vytváří, udržuje, spravuje a uvádí do provozu své assets
 - zajištění provozu systémů, dat, informací a bezpečnosti
- enterprise architecture
 - definuje celkové prostředí, do kterého zasahují jednotlivé produkty – zachycení cross systém aspektů
 - technické a organizační rámce, síťové prostředí, konfigurace, podpůrná infrastruktura → omezení architektury systémů
- business modeling
 - porozumění stávajícím (nezměněným) cílům, hranicím, procesům a strukturám organizace zákazníka
 - význam pro BPR, potřeba hlídat rozsah
- software process improvement
 - podpora analýzy, vytváření a používání sw procesů vedoucích k efektivitě
 - agile – průběžná aktivita (retrospektivy)