Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	ZS 2015/2016
Autoři **– jméno, příjmení, xname**	Tomáš Herout, xhert13 Michal Větrovec, xvetm05
Téma	Porovnání RUP disciplíny Test s odpovídajícími PA v CMMI-DEV v. 1.3 a procesy v ISO/IEC 12207
Datum odevzdání	31. 12. 2015

Abstrakt

Tato práce se zabývá porovnáváním v oblasti testování software. Je zde porovnávána disciplína Test metodiky RUP, procesní oblasti z metodiky CMMI-DEV a vybrané procesy standardu ISO/IEC 12207-2008.

Klíčová slova

RUP, test, testování, porovnání, software, ISO, ISO/IEC, CMMI, CMMI-DEV, metodika, metodika vývoje, standard, norma, vývoj software, test software

Obsah

1 Úvod

Testování je pro vývoj software velmi důležité. Neotestovaný software je plný potenciálních chyb. Zvláště pokud se jedná o komplexní systém, je náchylnost k chybovosti ještě větší. Neotestovaný produkt s chybami jednak snižuje reputaci podniku, která software vyrobila, a jednak zvyšuje náklady na opravy. Čím později jsou chyb odhaleny, tím nákladnější je jejich odstranění. V pozdějších stádiích vývoje, kdy je již produkt skoro hotov, je nákladnější (jak časově, tak peněžně) chyby opravit. Důvodem je daleko větší část, do které se musí zasáhnout a větší riziko vzniku chyb dalších. Proto je dobré s testováním při vývoji začít co nejdříve. V současnosti je vzrůstajícím trendem vývoj řízený testy.

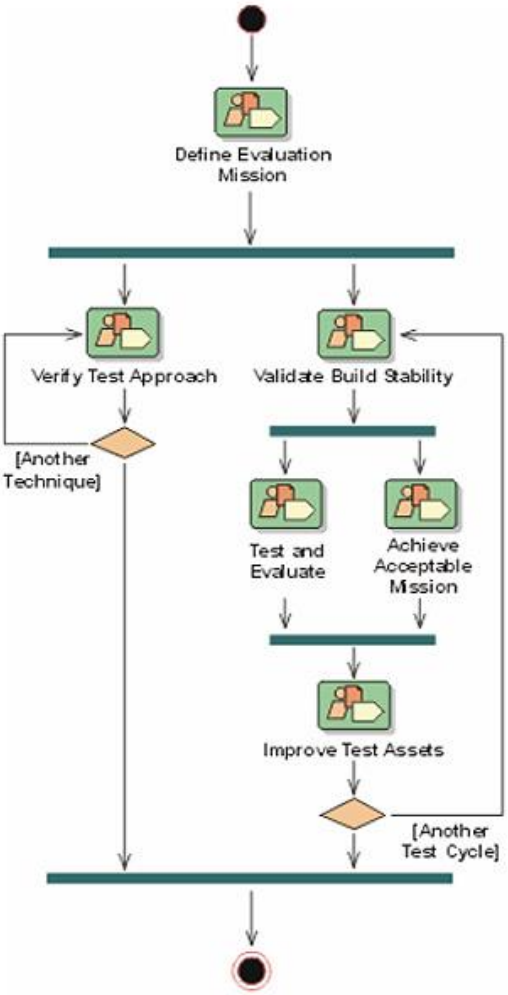
Naším úkolem je porovnat disciplínu Test metodiky RUP, odpovídající procesní oblasti v referenčním modelu CMMI-DEV a vybrané procesy standardu ISO/IEC 12207:2008. V práci nejprve obecně představíme všechny tři subjekty a následně popíšeme vybrané oblasti. Poté vybrané oblasti porovnáme a nalezené poznatky zaznamenáme.

2 Seznámení s RUP disciplínou Test

Pro disciplínu Test platí celá řada základních mechanismů, které jsou analyzovány níže. Pro RUP je typické přesné definování pracovních postupů, vstupů i výstupů aktivit a jednotlivých rolí.

2.1 Jasné vymezení disciplíny Test

RUP organizuje aktivity do podobných skupin nazvaných disciplíny. V tomto modelu je testování disciplína sama o sobě. Test disciplína se podílí na hlavním vývojovém procesu, je jak konzumentem, tak i producentem artefaktů a aktivit – tyto artefakty i aktivity jsou v RUP jasně definované. Instrukce pro testování v RUP definují artefakty, šablony, nástroje, pokyny, a checkpointy k použití (Ness a Thomas 2005). Obrázek níže ukazuje pracovní postup (workflow), který je pevně daný touto disciplínou.



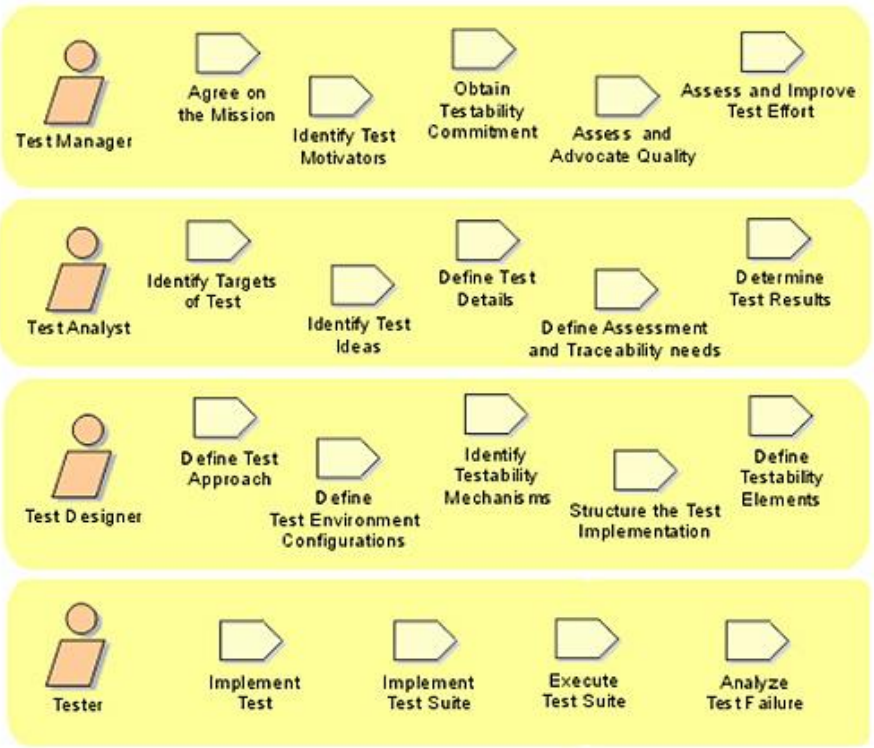
OBRÁZEK 1 – WORKFLOW DISCIPLÍNY TEST

Rozhraní mezi ostatními disciplínami a disciplínou Test je rovněž jasně stanovené, z čehož plyne, že artefakty i aktivity (veškerá dokumentace) jsou uzpůsobeny testování a není nutné je předělávat.

2.2 Role v disciplíně Test

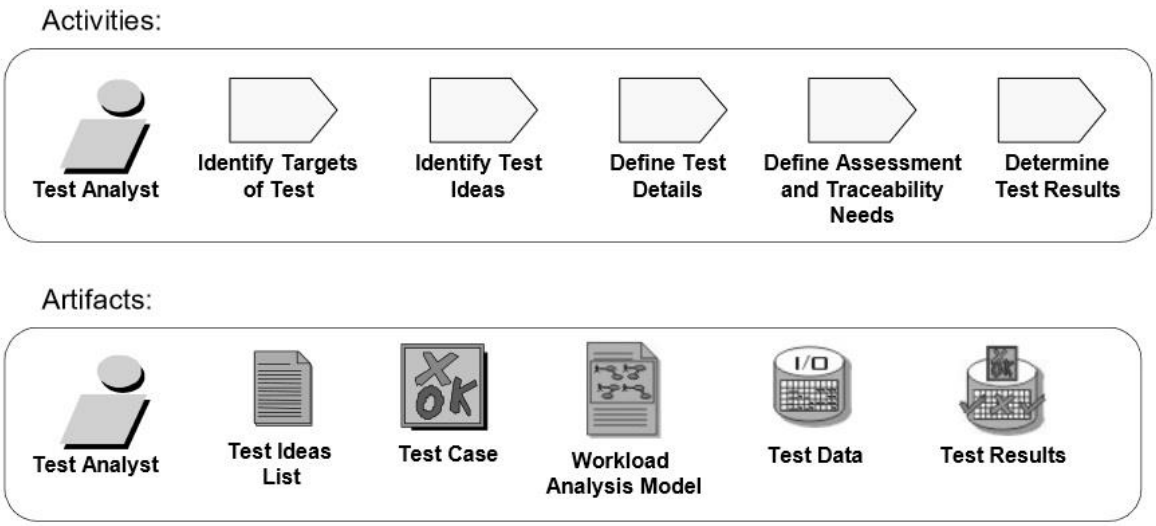
RUP nedefinuje jen jednu roli speciálně pro test disciplínu, ale čtyři, jak je znázorněno na dalším obrázku. Jeden tester může provádět od jedné po všechny čtyři role, ale role jsou jasně definovány, stejně jako odpovědnost za činnosti a artefakty. To znamená, že RUP disciplína Test může být dobře

škálovatelná z podnikání s jedním testerem na obrovskou firmu s členy testovacího týmu distribuovanými přes více kontinentů.



OBRÁZEK 2 – ROLE VČETNĚ AKTIVIT (NESS A THOMAS 2005)

Jednotlivé role pak mají kromě aktivit také definovány artefakty včetně detailního slovního popisu. Další obrázek je např. demonstrace analytika testů.



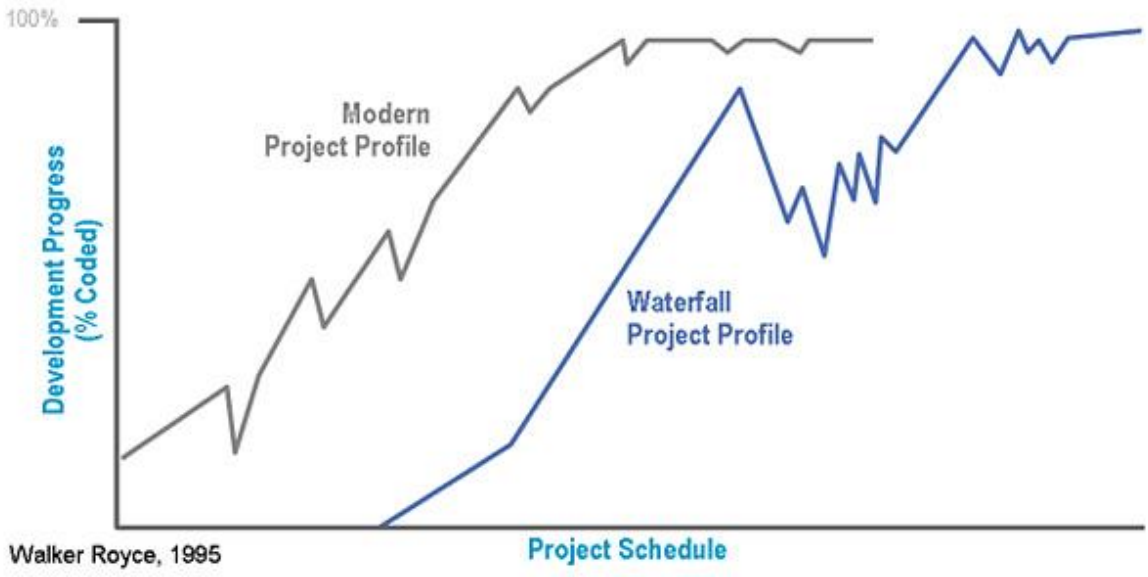
OBRÁZEK 3 – ČINNOSTI A ARTEFAKTY ANALYTIKA TESTŮ (IBM CORP 2006)

2.3 Použití iterativního vývojového přístupu

Dříve byl software jednodušší, méně komplikovaný a týmy, co měly na starosti jeho vývoj, byly menší. Používal se vodopádový model. Ten nefunguje tak dobře u větších projektů, protože se testuje vše najednou nakonec (Ness a Thomas 2005).

S iterativním přístupem tým testerů dostane menší porci software k otestování – dlouho před tím, než je celý projekt hotový. Např. u devítiměsíčního projektu se praktikují šestitýdenní iterace a šest oddělených vydání pro otestování částečných funkcionalit, což dává více příležitostí pro testery ohodnotit finální produkt (Ness a Thomas 2005).

Jak můžete vidět na obr. níže, iterativní přístup dává příležitost dělat menší opravy dříve, takže náklady na opravy se snižují. Redukuje se také riziko, zvyšuje kvalita a je možnost dodat produkt dříve.



OBRÁZEK 4 – VODOPÁDOVÝ VS. ITERATIVNÍ PŘÍSTUP

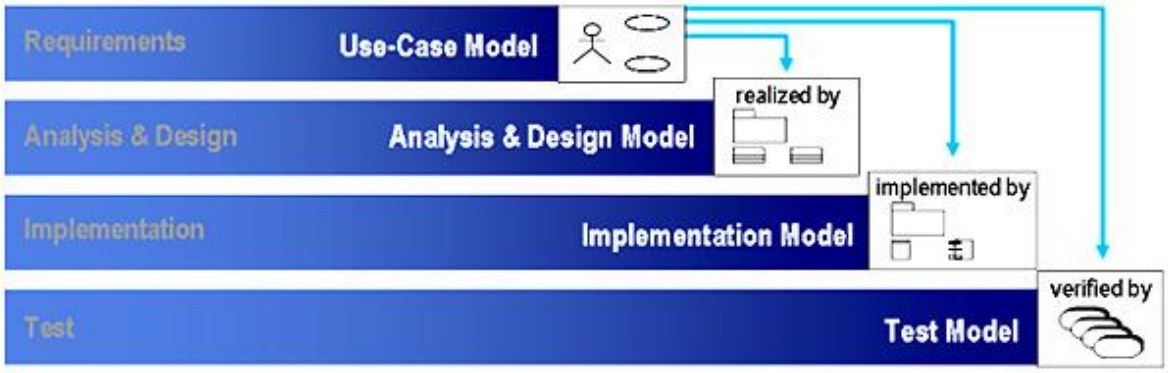
2.4 Neustálé ověřování kvality

Této praktiky (ve zdrojovém článku je dokonce použit termín best practice) je dosaženo definováním tzv. checkpointů (záchytných bodů) pro každý artefakt, aktivitu nebo část dodání v procesu. RUP dává jasný návod na to, jak vykonat každou aktivitu, což velmi napomáhá konzistenci mezi projekty a každý člen týmu zná své cíle (Ness a Thomas 2005).

Díky tomuto systému checkpointů může každý zainteresovaný hodnotitel ohodnotit kvalitu bez toho, aby chodil na formální meetingy.

2.5 Použití use case pro řízení vývoje

Požadavky na funkce jsou primárně zachyceny ve formě Use-case diagramů (případů užití). Design a implementace se zaměřuje na dosažení všeho popsaného v těchto diagramech. Dodávka na konci iterace je zaměřena na zákazníkem vnímanou hodnotu a to bere v potaz i celý tým, který má právě testování na starosti (Ness a Thomas 2005). Testovací tým hodnotí případy užití, nikoli interní komponenty.



OBRÁZEK 5 – JAK USE-CASE MODEL ŘÍDÍ VŠECHNY FÁZE RUP (NESS A THOMAS 2005)

2.6 Plánování implementace na základě rizika

Nejrizikovější funkcionality by měly být implementovány nejdříve. Díky tomu se změny ohledně termínů projeví velmi brzy a je snadnější pro celou firmu se jim přizpůsobit.

2.7 Strategické řízení změn

Změny v projektu, kdy vývojáři přidávají velké funkcionality těsně před vydáním, jsou velmi špatné. Správná strategie řízení změn dává také testovacímu týmu možnost komunikace s vývojáři při nalezení problému, což vede k méně neshodám mezi těmito dvěma týmy, lepší volbě priorit a méně promarněnému času testováním něčeho, co např. ještě nemuselo být opraveno (Ness a Thomas 2005).

2.8 Použití adekvátních procesů

Někdy u malých projektů by bylo možné říci, že je toto vše „zbytečné“ a lze použít ad hoc neformální proces. RUP však není o tvorbě nepotřebných aktivit, vše musí mít své místo a vše musí přinášet hodnotu. U velkých projektů je nutné mít formální procesy, aby byla komunikace jasná a nevznikala nedorozumění. RUP je škálovatelný na všechny velikosti projektů, od jednočlenných týmů po obrovské organizace (Ness a Thomas 2005).

2.9 Shrnutí RUP disciplíny Test

RUP je kompletní vývojový procesní rámec, který podporuje kvalitní, on-time, on-budget, dobře otestované softwarové projekty. Přijetí a používání RUP usnadňuje život nejen těm, kteří jsou závislí na výsledcích od testovacího týmu, ale také samotným členům testovacího týmu.

3 CMMI-DEV v.1.3

3.1 Obecně

CMMI-DEV je referenční model procesů, které pokrývají celý životní cyklus softwarového produktu (od návrhu, přes vývoj a následnou údržbu). Proces je v CMMI modelu definován jako sada vzájemně provázaných činností, které transformují vstupy na výstupy pro dosažení definovaného účelu (Buchalcevová 2015).

CMMI dále udává různé úrovně způsobilosti a zralosti, což lze vidět na tabulce níže.

úroveň	kontinuální reprezentace úrovně způsobilosti (capability levels)	stupňovitá reprezentace úrovně zralosti (maturity levels)
0	neúplný (Incomplete)	
1	vykonávaný (Performed)	úvodní (Initial)
2	řízený (Managed)	řízená (Managed)
3	definovaný (Defined)	definovaná (Defined)
4		kvantitativně řízená (Quantitatively Managed)
5		optimalizovaná (Optimizing)

OBRÁZEK 6 – ÚROVNĚ ZPŮSOBILOSTI A ZRALOSTI PROCESŮ

3.2 Odpovídající procesní oblasti

Model CMMI-DEV definuje 22 procesních oblastí. Jako oblasti korespondující s obsahem RUP disciplíny Test jsme vybrali ty v následující tabulce. Dále by se dalo spekulovat o zařazení oblasti Risk Management, protože v RUP je přímo zmíněno plánování implementace založené na riziku, ale tato oblast zasahuje hodně do managementu, takže jsme zůstali pouze u vybraných čtyř.

Verification a Validation jsou podobné, ale zaměřují se na rozdílné problémy. Verifikace zajišťuje, že je SW budován správně a validace oproti tomu zajišťuje, že firma buduje správnou věc.

KATEGORIE	PROCESNÍ OBLAST
Engineering	Validation (VAL)
Engineering	Verification (VER)
Project Management	Quantitative Project Management (QPM)
Support	Process and Product Quality Assurance (PPQA)

3.2.1 Validation (VAL)

Validace je procesní oblast úrovně zralosti 3. Účelem této oblasti je dokázat, že produkt nebo jeho komponenty splňují zamýšlenou funkčnost v cílovém prostředí (CMMI Product Team 2010). Základní myšlenky validace jsou (CMMI Product Team 2010):

- Příprava validace je řízena.
- Výběr produktů a komponentů k ověření a výběr správných metod, které mají být k tomuto použity.
- Zřízení a údržba prostředí potřebného pro ověřování.
- Zřízení a údržba procedur a kritérii pro ověřování.
- Samotná analýza výsledků validace.

Pod prostředím si lze představit nástroje pro testování, zkušený personál, zařízení, nástroje pro záznam a celou řadu dalších (CMMI Product Team 2010).

Tato PA zmiňuje ověřování uživatelského rozhraní, uživatelských manuálů, přístupových protokolů, dokumentace, požadavků na produkty/komponenty a další (CMMI Product Team 2010).

Kromě tohoto zmiňuje i celou řadu ověřovacích metod, jako např. diskuze s koncovými uživateli, použití prototypů, demonstrace funkcí uživatelům apod. (CMMI Product Team 2010).

Mezi příbuzné oblasti patří Verifikace, Requirements Development a Technical Solution (CMMI Product Team 2010).

3.2.2 Verification (VER)

Verifikace je PA s úrovní zralosti 3 (stejnou jako u validace). Základní myšlenky se takřka zcela kryjí s těmi u validace, proto shrnu pouze nejdůležitější body:

- Správný výběr produktů pro verifikaci a verifikačních metod, které budou použity.
- Samotné ověření produktů oproti jejich požadavkům.
- Analýza výsledků.

Jako příbuzné oblasti jsou zmíněny Validace, Requirements Development a Requirements Management (CMMI Product Team 2010).

3.2.3 Quantitative Project Management (QPM)

QPM je PA úrovně zralosti 4. Tato oblast zahrnuje zejména následující (CMMI Product Team 2010):

- Vytvoření a udržování kvality projektu a cílů sledovaných v procesech.
- Tvorba jasně definovaných procesů.
- Volba podprocesů a atributů rozhodujících pro výkon, pomáhajících k dosažení cílů projektu a patřičné kvality procesů projektu.
- Výběr měřítek a analytických technik.
- Monitoring výkonnosti procesů/podprocesů.
- Analýzy, statistické a kvantitativní metody pro dosažení objektivního hodnocení produktu.

Z této oblasti benefituje nejen testování, ale i další oddělení společnosti – identifikace rizik, průzkum trhu, průzkum nových technologií atd. (CMMI Product Team 2010).

3.2.4 Process and Product Quality Assurance (PPQA)

Jedná se o podpůrnou procesní oblast úrovně zralosti 2. Zahrnuje následující aktivity:

- Objektivní hodnocení provedených procesů a pracovních produktů podle popisu procesů, standardů a procedur.
- Identifikace a dokumentace nalezených chyb.
- Poskytování zpětné vazby managementu, sledování stavu problému.
- Ujištění se o řešení problému.
- Vedení záznamů o testování.

Tato oblast dává i několik příkladů, jak provést objektivní hodnocení, např. pomocí formálních auditů organizovaných nějakou nezávislou externí firmou. Zdůrazňuje se zde také důležitost školení testerů. Testovat by se mělo začít v brzkých fázích projektu (CMMI Product Team 2010).

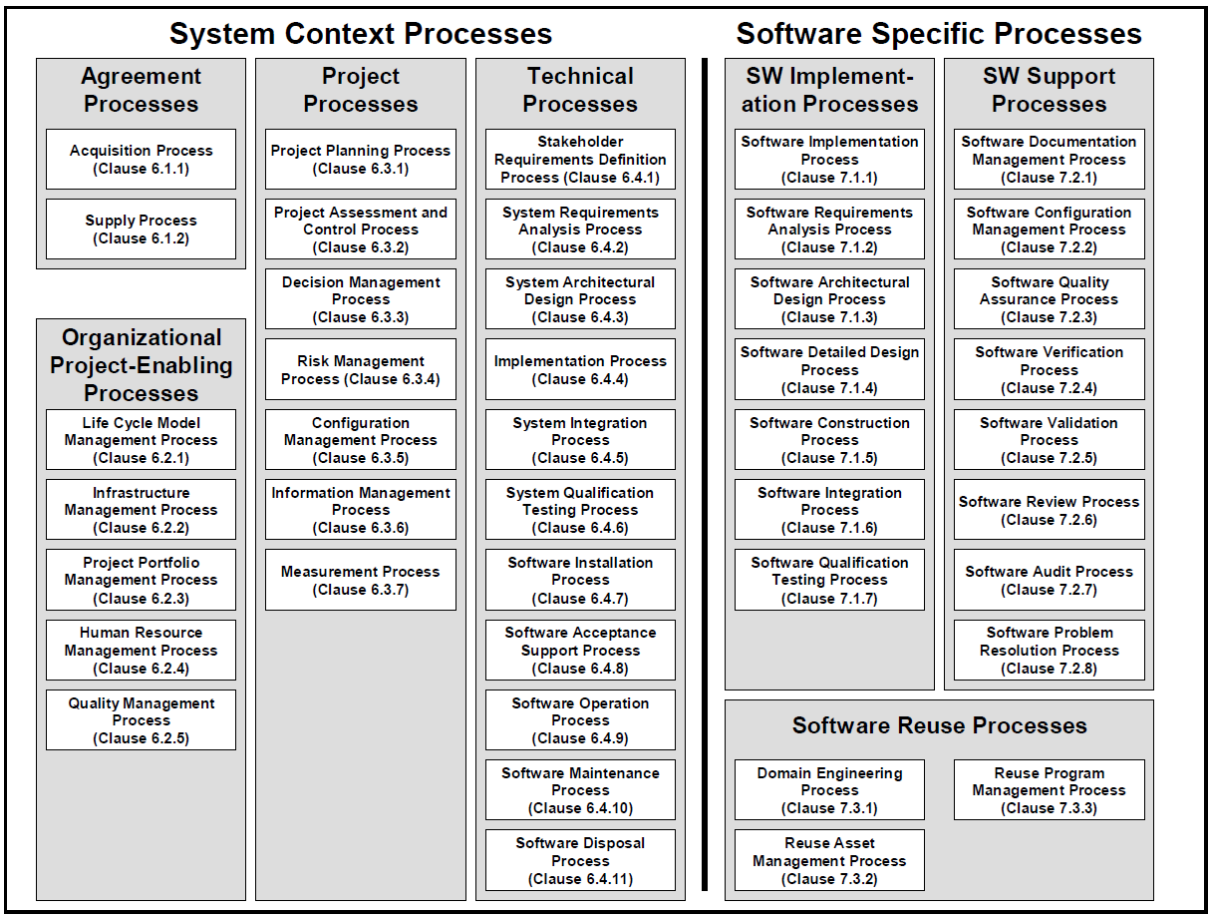
V této procesní oblasti je přímo zmíněno, že Verifikace je související oblast, a (nejen) proto jsme ji vybrali.

4 ISO/IEC 12207

4.1 Obecně

Tento standard přímo nepředepisuje přesně stanovené postupy nebo workflow. Cílem je poskytnout soubor procesů ke komunikaci mezi dodavateli a objednateli software a mezi dalšími zájmovými stranami. (SPONSOR, 2008)

Procesy jsou rozděleny na dvě hlavní skupiny a ty pak na několika dalších podskupin. Těmi hlavními skupinami jsou **System Context Processes** a **Software Specific Processes**. System Context Processes jsou procesy samostatného software nebo celého systému aplikací, zatímco Software Specific Processes se zabývá procesy služby nebo software produktu, který je součástí většího celku. Rozdělení se všemi procesy, které ISO 12207 obsahuje, můžeme vidět na následujícím obrázku.



OBRÁZEK 7 – SKUPINY PROCESŮ ŽIVOTNÍHO CYKLU (SPONSOR, 2008)

Každý proces je popsán samostatně a má 4 části:

- Název,
- Účel,
- Výstupy,
- Činnosti a úkoly

Vše je popsáno ve velmi obecné rovině. Role nejsou přesně definovány a v mnoha případech nejsou u procesu definovány vůbec. V námi vybraných procesech pro toto srovnání jsme nalezli cca 5 rolí, což není mnoho. Velmi často standard pouze doporučuje, aby byla stanovena odpovědná autorita, bez dalších přesnějších specifikací, kdo by to měl být nebo co by mělo být jeho úkolem.

Rovněž ani výstupy nejsou popsány dopodrobna. Je pouze uvedeno, jaký výstup by měl vzniknout, ale co přesně by mělo být obsahem nebo jaká by měla být struktura, tento standard nezachycuje.

V popisu je také často odkazováno, a to buď na části téhož standardu, nebo standardy jiné.

4.2 Odpovídající procesy

Protože testování je součástí nejen vývoje softwarové komponenty nebo části systému, ale i systému jako celku či celé aplikace, vybrali jsme procesy z obou hlavních skupin. Z procesů, které jsou v normě definovány, jsme vybrali následujících 5. Kterých 5 bylo vybráno, shrnuje následující tabulka:

KATEGORIE	SUBKATEGORIE	PROCESNÍ OBLAST
System life cycle processes	Technical Processes	System Qualification Testing Process
Software Specific Processes	SW Implementation Processes	Software Qualification Testing Process
Software Specific Processes	SW Support Processes	Software Quality Assurance Process
Software Specific Processes	SW Support Processes	Software Verification Process
Software Specific Processes	SW Support Processes	Software Validation Process

4.2.1 System Qualification Testing Process

Cílem tohoto procesu je ujistit se, že každý požadavek na systém byl splněn a že systém je připraven k dodání. (SPONSOR, 2008)

Jako jedny z výstupů je doporučeno vytvořit:

- Vytvořit sadu hodnotících kritérií a s jejich pomocí systém otestovat
- Výsledky testů zaznamenat

Proces doporučuje provést následující:

- Je doporučeno testování způsobilosti, zdokumentovat výsledky
- Test pokrytí požadavků jako jeden z testů test pokrytí požadavků
- Měl by proběhnout audit

Z rolí je zmíněn pouze vývojář a to u auditu. Vývojář by měl audit podporovat a pokud se uskuteční, měl by jeho skončení připravit software pro instalaci a akceptaci. (SPONSOR, 2008)

Na konci popisu se píše, že tento proces může být použit i v procesech Software Verification Process nebo Software Validation Process. I to je důvod, proč jsme procesy Verification a Validation zahrnuli do našeho srovnání.

4.2.2 Software Qualification Testing Process

Proces má potvrdit splnění požadavků na jednotlivé integrované části software. (SPONSOR, 2008)

Průběh tohoto procesu je velmi podobný, téměř identický s procesem System Qualification Testing Process. Mimo jiné by měl zahrnovat:

- Zvolení kritérií a jejich použití při testování
- Zaznamenání výsledků testování
- Vytvoření regresní strategie
- Provedení testování způsobilosti
- Mělo by být otestováno splnění každého požadavku na software
- Provést test pokrytí požadavků a výsledky porovnat s očekáváním

Role je zde opět pouze jedna je jí implementátor. Ten průběžně aktualizuje dokumentaci a hodnotí návrh, kód, testy a jejich výsledky.

I zde by měla jediná zmíněná role, tedy implementátor, podporovat audit. Jeho výsledky by měl implementátor dokumentovat. (SPONSOR, 2008)

Stejně jako u System Qualification Testing Process lze tento proces použít i v procesech Software Verification Process nebo Software Validation Process.

4.2.3 Software Quality Assurance Process

Účelem procesu Software Quality Assurance je ujistit se, že výrobky a procesy jsou v souladu s předdefinovanými předpisy a plány. (SPONSOR, 2008)

Mělo by se provést toto:

- Vyvinuta strategie pro řízení kvality
- Jsou zaznamenány problémy a nesoulad s požadavky
- Měl by vzniknout plán pro činnosti a úkoly a měl by být zdokumentován a udržován
- Zajistit, že softwarové produkty jsou v souladu se smlouvami a drží se plánů, plně vyhovují požadavkům
- Zajistit, že stejně jako produkty i procesy životního cyklu jsou v souladu se smlouvami a drží se plánů

Software Quality Assurance by měl být koordinován spolu s těmito dalšími procesy:

- Software Verification
- Software Validation
- Software Review
- Software Audit

Osoby odpovědné za tento proces by měly dostat patřičnou svobodu a potřebné zdroje a stát se autoritami.

Z rolí jsou zde zmínění subdodavatel a objednatel („acquirer“). Dále jsou jen obecně zmíněné odpovědné osoby a zaměstnanci, tyto dvě skupiny ale nepovažujeme za role.

4.2.4 Software Verification Process

Úkolem Verifikace je potvrdit, že každý projekt, produkt nebo služba odpovídá stanoveným požadavkům. (SPONSOR, 2008)

Mezi výstupy zde počítáme:

- Vytvořena strategie pro verifikaci
- Identifikována kritéria pro verifikaci
- Zaznamenány nedostatky

- Výsledky jsou dány k dispozici zákazníkovi a zájmovým stranám

Mělo by být rozhodnuto, jestli projekt zasluhuje verifikaci. Požadavkům projektu by měla být přidělena důležitost. Pokud projekt zasluhuje verifikaci, měl by být implementován proces verifikace k ověření software. Pokud je proces implementován, měl by vzniknout verifikační plán na základě komplexity a rozsahu projektu. Bylo by dobré určit, které produkty a činnosti vyžadují verifikaci a zahrnout je do plánu.

Problémy nalezené při běhu verifikačního plánu jsou součástí Software Problem Resolution Process, který ovšem není předmětem této práce, proto jej nebudeme nijak dále popisovat.

Samotný průběh verifikace má 5 částí:

- Verifikace požadavků
- Verifikace návrhu
- Verifikace kódu
- Verifikace integrace
- Verifikace dokumentace

Ke každé části norma navrhuje vlastní kritéria, která by měla být zvážena k použití.

V procesu verifikace je zmíněna jediná role „acquirer“. Acquirer by měl mít k dispozici výsledky verifikačních činností.

4.2.5 Software Validation Process

Účelem je potvrdit, že jsou splněny požadavky na specifické použití produktu. (SPONSOR, 2008)

Popis tohoto procesu je velmi podobný předcházejícímu procesu Verification. Podobně jako v předchozím procesu by mělo být rozhodnuto, jestli projekt zasluhuje validaci. Pokud ano měl by být implementován a dokumentován validační proces. Norma dále doporučuje vytvořit validační plán, který by kromě jiného měl obsahovat:

- Předměty validace
- Validační úkoly
- Zdroje, naplánování a odpovědnosti

Nalezené problémy jsou opět součástí procesu Software Problem Resolution Process.

Během samotné validace probíhá:

- Příprava vybraných požadavků a specifikací pro analýzu testování
- Testování jednotkových, krajních a rozsahových vstupů
- Testování chyb
- Testování uživatelské přívětivosti
- Ověření, že software splňuje zamýšlený účel

V procesu validace jsou zmíněny dvě role a to „conductor“ a „acquirer“.

5 Porovnání

V této kapitole se zaměříme na samotné porovnání. Rozdělena je na dvě části. V části první uvádíme, co mají zkoumané subjekty společného a kde se shodují a v části druhé jsou pak rozdíly zobrazené v tabulce.

5.1 Co je stejné?

Při procházení subjektů k porovnání, jsme hledali ty aspekty, které jsou pro všechny tři zcela nebo alespoň z velké části společné. Oblasti týkající se testování jsme nalezli u všech.

Podařilo se nám identifikovat tyto společné oblasti:

- Rozhraní mezi disciplínami je jasně definované
    - Není nutné předělávat artefakty, aktivity ani dokumenty
    - S testováním se počítá celou dobu
- Všechny porovnávané subjekty se v první řadě zaměřují na dodání návodu nebo doporučení s cílem vytvářet kvalitní software
- Komunikace mezi testery a developery (managementem)
- S testováním začít brzy
- Vztahy k ostatním disciplínám – Na ostatní oblasti je odkazováno nebo upozorněno, které procesy mohou být sdílené a kde je například třeba kooperovat s jinými procesy

5.2 Odlišnosti

Následující tabulka zachycuje odlišnosti, které jsme u zkoumaných subjektů nalezli. Každý řádek představuje jedno kritérium.

RUP Test	CMMI	ISO/IEC 12207
Definuje role	Nedefinuje role	Definuje malý počet rolí
Artefakty a aktivity (včetně detailního popisu vstupních/výstupních art.)	Pracovní produkty a podpraktiky (work products and subpractices) – jen výpis	Výstupy, činnosti a úkoly ve velmi obecné rovině
Lepší konzistence	Přesah některých PA do jiných, nejasné ohraničení	Procesy přehledně rozděleny do skupin a podskupin, odkazují mezi sebou a mezi dalšími ISO standardy
Nezdůrazňuje objektivitu při testování	Zdůrazňuje objektivitu při testování (zejména v PPQA)	Nezdůrazňuje objektivitu při testování
Nezdůrazňuje ponechávání záznamů o testování	Zdůrazňuje ponechávání záznamů o testování	Doporučuje vést záznamy a dokumentovat
Nezmiňuje kvantitativní a statistické metody pro testování	Zmiňuje kvantitativní a statistické metody pro testování	Nezmiňuje kvantitativní a statistické metody pro testování
Nezmiňuje monitoring	Monitoring výkonnosti	Nezmiňuje monitoring
Iterativní přístup	-	-
Use-case driven	-	-
Checkpointy	-	-
Zohlednění faktoru rizika	-	Riziko nezohledňuje
Jasně definované workflow	-	Pouze výčet činností, které by měly být provedeny
Detailní popis postupů (guidance)	-	Obecně definované úkoly

6 Závěr

Naším úkolem bylo představit a porovnat disciplínu Test metodiky RUP, odpovídající procesní oblasti v referenčním modelu CMMI-DEV a vybrané procesy standardu ISO/IEC 12207:2008.

Jednotlivé subjekty a jejich vybrané části byly představeny v kapitolách 2, 3 a 4 a jejich podkapitolách. Samotné porovnání bylo provedeno v kapitole 5 a jejích částech.

Celkově jsme nalezli více rozdílností než společných vlastností, což se při různých zaměřeních jednotlivých subjektů dá lehce pochopit. RUP je konkrétní metodika určená pro vývoj software, vše proto stanovuje velmi konkrétně a do detailů. Naproti tomu ISO je obecný standard, proto je vše velmi obecné a spíše formou doporučení. CMMI-DEV jako referenční model některé oblasti definuje obecněji, některé konkrétněji.

Cíl naší práce – porovnání – považujeme za splněný.

7 Literatura

NESS, Pete a Lee THOMAS. 2005. *The Rational Unified Process for testers: Building in quality from the start*. IBM developerWorks [online]. [cit. 2015-11-29]. Dostupné z: http://www.ibm.com/developerworks/rational/library/04/r-3239/

IBM Corp. 2006. *RUP for Small Projects*. [online]. [cit. 2015-12-4]. Dostupné z: https://kitscm.vse.cz/RUP/SmallProjects/index.htm

CMMI Product Team. 2010. *CMMI for Development, Version 1.3*. [online]. [cit. 2015-12-4]. Dostupné z: http://www.sei.cmu.edu/reports/10tr033.pdf

SPONSOR, Software. *Systems and software engineering software life cycle processes*. 2. vyd. Geneva: ISO/IEC-IEEE, 2008 [cit. 2015-12-3]. ISBN 0-7381-5664-7.