Implementační balíček
Konstrukce a Jednotkové testování
Basic Profil

Poznámky:

Tento dokument je duševním vlastnictvím autora. Nicméně informace obsažené v tomto dokumentu jsou volně dostupné. Distribuce všech částí tohoto dokumentu je povolena pro nekomerční použití na tak dlouho, jak je uvedeno v následujícím právním upozornění:

Komerční užití tohoto dokumentu je přísně zakázáno. Tento dokument je distribuován za účelem posílení výměny technických a vědeckých informací.

Tento materiál je koncipován na bázi "as-is". Autor nedává žádnou záruku jakéhokoli druhu, ať už výslovně uvedené nebo předpokládané vhodnosti pro daný účel, prodejnost, výlučnost či jiné výsledky získané z použití tohoto materiálu.

Procesy popsané v tomto implementačním balíčku nejsou určeny k vyloučení nebo odrazení používání dalších procesů, které mohou najít využití ve VSE.

Autor	ANA VAZQUEZ – 5th level (México)
Redakce	ANA VAZQUEZ – 5th level, (México) C. Y. LAPORTE – École de technologie supérieure (ÉTS), (Canada)
Vytvořeno	28 duben 2009
Poslední aktualizace	6 října 2012
Verze	0.4

Historie verzí

Datum (yyyy-mm-dd)	**Verze**	**Popis**	**Autor**
2009-04-28	0.1	Vytvoření dokumentu	Ana Vázquez
2009-08-14	0.2	Revize	Claude Laporte
2009-10-15	0.3	Aktualizace části 5 a úprava SI 4.1 podúkolů	Ana Vázquez
2010-09-21	0.4	Změna úkolu a podúkolu uvedené v SI 4.3 k dosažení konzistence s poslední 29110-5 a revize sekce 9.	Ana Vázquez

Zkratky/Akronymy

Zkratky/Akronymy	**Definice**
DP	Implementační balíček – soubor artefaktů vyvinutých s cílem usnadnit implementaci souboru postupů, vybraného rámce ve velmi malém podniku.
VSE	Velmi malý podnik – podnik, organizace, oddělení nebo projekt mající do 25 lidí.
VSEs	Velmi malé podniky
UI	Uživatelské rozhraní
TL	Technický vedoucí
AN	Analytik
DES	Návrhář
PR	Programátor

Obash

1. Technický popis

Účel tohoto dokumentu

Implementační balíček podporuje Basic profil, jak je definován v ISO/IEC 29110 v části 5-1-2: Manažerská a technická příručka. Basic profil je jeden ze skupiny obecných profilů. Obecná skupina profilů je složena ze 4 profilů: Entry, Basic, Intermediate a Advanced. Skupina profilů je použitelná pro velmi malé podniky, které nevyvíjí kritický software. Skupina obecných profilů nevyžaduje žádnou specifickou aplikační doménu.

Implementační balíček je sada artefaktů vyvinutých s cílem usnadnit implementaci sady praktik ve velmi malých podnicích. Implementační balíček není procesním referenčním modelem (tj. není předepisující). Prvky typické pro Implementační balíček jsou: popis procesů, činností, úkolů, rolí a produktů, šablon, kontrolních seznamů, vzorů, referencí a odkazů na standardy, modelů a nástrojů.

Obsah tohoto dokumentu je pouze informativní.

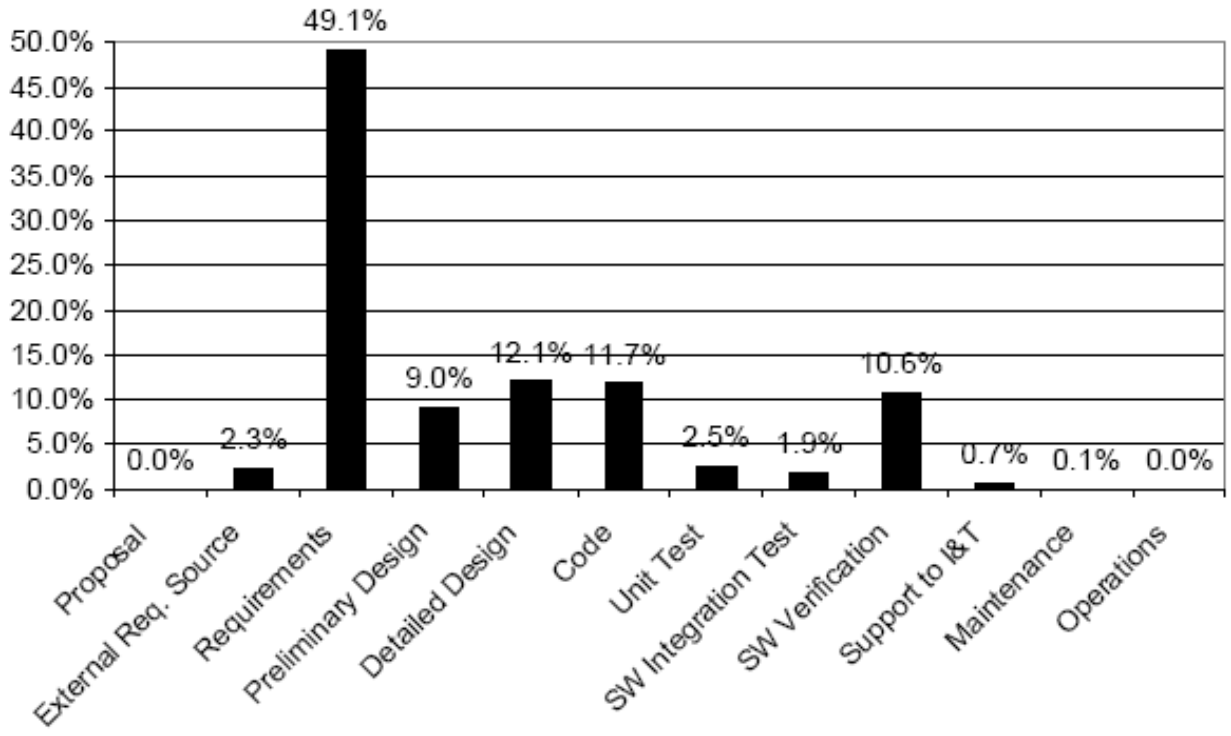
Tento dokument byl vytvořen Annou Vazquez z 5th level (Mexico) při její účasti na ISO JTC1/SC7/WG24.

Proč je Konstrukce a Jednotkové testování důležité?

Není potřeba vysvětlovat do velkých detailů, proč je Konstrukce a Jednotkové testování důležité. Jsou to jediné činnosti, které se provádí v každém projektu, který vytváří software, na rozdíl od jiných činností, jako činnosti projektového řízení, které jsou někdy přeskočeny nebo neprovedeny úplně. Konstrukce není jen prováděna, ale může být doplněna o činnosti Analýzy a Návrhu dle různých agilních přístupů jako Extrémní programování a Scrum.

Konstrukce je ve většině případů činností, která zabere většinu času Softwarového projektu. Její provedení má významný dopad na náklady projektu a Plán projektu.

Co se týče kvality, okolo 12% chyb, jak je znázorněno na obrázku 1, jsou objeveny během Konstrukce. Ty by měly být odstraněny prostřednictvím jednotkových testů.



Obrázek 1 původy softwarových chyb (Selby 2007)

Konstrukční činnosti by měly být plánované v průběhu fáze přípravy projektu. Konstrukční činnosti by měly být popsány v Plánu projektu.

2. Definice

V této části najde čtenář dva soubory definic. První soubor definuje pojmy používané ve všech Implementačních balíčkách, tj. Obecné pojmy. Druhý soubor tvoří řada termínů použitých v tomto Implementačním balíčku, tj. specifické a konkrétní termíny.

Obecné termíny

Proces: soubor vzájemně souvisejících nebo vzájemně působících činností, které přeměňují vstupy na výstupy [ISO/IEC 12207].

Činnost: soubor soudržných, kompaktních úkolů procesu [ISO/IEC 12207].

Úkol: požadované, doporučené nebo povolené akce, s cílem přispět k dosažení jednoho nebo více výsledků procesu [ISO / IEC 12207].

Podúkol: pokud je úkol složitý, je rozdělen do podúkolů.

Krok: v Implementačním balíčku je úkol rozdělen ve sledu kroků.

Role: definované funkce, které mají být provedeny členem projektového týmu, jako například testování, plnění, kontrola, kódování [ISO/IEC 24765].

Produkt: dílčí informace nebo dodávka, která může být získána jedním nebo více úkoly (např. Projektová dokumentace, zdrojový kód).

Artefakt: informace, které nejsou uvedeny v ISO/IEC 29110, část 5, ale mohou pomoci velmi malým podnikům v průběhu realizace projektu.

Konkrétní termíny

Komponenta: soubor funkčních služeb softwaru, který po implementaci představuje správně definovaný soubor funkcí a je rozlišitelný prostřednictvím jedinečných jmen [ISO/IEC 29881:2008].

Chyba: problém, který není-li opraven, může způsobit selhání aplikace nebo podávat nesprávné výsledky [ISO/IEC 20926].

Přepracování: opatření přijatá na upravení chybné nebo nevyhovující komponenty do souladu s požadavky a specifikacemi [PMBOK].

Trasovatelnost: míra, do jaké vztah může být stanovený mezi dvěma nebo více produkty vývojového procesu, zejména produkty mající předchůdce, nástupce nebo vztah podřízený – nadřízený mezi sebou **[IEEE 1233-1998].**

Jednotkový test: testování jednotlivých postupů a modulů vývojáři nebo nezávislými testery [ISO/IEC 24765].

3. Vztahy k ISO/IEC 29110

Tento Implementační balíček zahrnuje činnosti související s Konstrukcí a Jednotkovým testováním ISO technické zprávy ISO/IEC 29110 část 5-1-2 pro velmi malé podniky (VSEs) – Basic Profile [ISO/IEC29110].

Konstrukční činnosti by měly být plánované v průběhu činnosti Plánování projektu. Konstrukční činnosti by měly být popsány v Plánu projektu. Pokud se tak nestane, měl by Projektový manažer vykonat tuto činnost před zahájením Konstrukce. (viz. Implementační balíček Projektové řízení)

V této části čtenář najde seznam činností, úkolů a rolí procesu Implementace softwaru (SI) z části 5, které jsou přímo spojené s tímto tématem. Toto téma je popsáno detailně v následující části.

- **Proces:** 7[1] Implementace softwaru
- **Činnost:** 7.7.7.2 Analýza softwarových požadavků
- **Úkoly a role:**

Úkoly	**Role**[2]
SI.2.2 Dokumentovat nebo aktualizovat Specifikaci požadavků. Identifikovat a konzultovat informační zdroje (zákazníci, uživatelé, předchozí systémy, dokumenty, apod.) s cílem získat nové požadavky. Analyzovat identifikované požadavky pro definování rozsahu a proveditelnosti. Generovat nebo aktualizovat Specifikaci požadavků.	Analytik, Zákazník

- **Činnost:** 7.7.1.4 Konstrukce softwaru
- **Úkoly a role:**

Úkoly	**Role**[3]
SI.4.1 Přidělit úkoly členům pracovního týmu vztahující se k jejich roli dle aktuálního *Plánu projektu.*	Technický vedoucí,

[1] Tato čísla odkazují na procesy, činnosti, úkoly ISO/IEC 29110 Part 5-1-2

[2] Role jsou definované v následující sekci. Role jsou také definované v ISO/IEC 29110 Part 5-1-2

	Programátor
SI.4.3 Konstruovat nebo aktualizovat *Softwarové komponenty* na základě detailní části *Softwarového návrhu.*	Programátor
SI.4.4 Navrhnout nebo aktualizovat jednotkové testy a použít jich k ověření, že S*oftwarové komponenty* implementují detailní *Softwarový návrh.*	Programátor
SI.4.5 Opravit nalezené chyby až do úspěšného proběhnutí jednotkového testu.	Programátor
SI.4.6 Do Záznamu trasovatelnosti zaznamenat vytvořené či modifikované Softwarové komponenty.	Programátor

4. Popis procesů, činností, úkolů, kroků, rolí a produktů

Proces: 7 Implementace softwaru

Činnost: 7.7.7.2 Analýza softwarových požadavků

Úkoly	Role[4]
SI.2.2 Dokumentovat nebo aktualizovat Specifikaci požadavků. Identifikovat a konzultovat informační zdroje (zákazníci, uživatelé, předchozí systémy, dokumenty, apod.) s cílem získat nové požadavky. Analyzovat identifikované požadavky pro definování rozsahu a proveditelnosti. Generovat nebo aktualizovat požadavky specifikace.	Analytik, Zákazník

Tento úkol souvisí s následujícími podúkoly:

- Vybrat standard uživatelského rozhraní
- Definovat standardy pro Konstrukci

Podúkol: Vybrat standard uživatelského rozhraní
Poznámka: Cíl tohoto podúkolu mohl být už dosažen, proto nemusí být požadován.

Cíl:	Vybrat standardizované rozhraní pro koncového uživatele
Zdůvodnění:	Požadavky na použitelnost jsou přímo předmětem těchto podúkolů. Tyto požadavky by měly být dohodnuty v rámci Specifikace požadavků, nicméně ve většině případů nejsou přímo uvedeny, ale jsou vždy očekávány. Standardy uživatelského rozhraní by měly být použity ve všech komponentách, které UI má. Jsou očekávány, i když nejsou specifikovány; když se tým rozhodne nepoužívat je, bude to viditelné pro koncové uživatele; budou vnímány jako nedostatečné, chybějící koordinace mezi projektovým týmem a software bude kvalifikována jako nekvalitní dokonce i když bude funkcionalita správně implementována. Použití standardu by mělo být požadováno a mohlo by tak být ušetřeno mnoho oprav.

[4] Role jsou definované v další části. Role jsou také definované v ISO/IEC 29110-5.1

	Jak bylo uvedeno v 29110-5, uživatelské rozhraní by mělo být navrženo v činnosti Softwarová architektura a Detailní návrh, nicméně je toto často přeskakováno. V tomto případě se může problém vyřešit provedením tohoto podúkolu.
Role:	Projektový manažer
	Technický vedoucí
Artefakty:	Specifikace uživatelského rozhraní
Kroky:	1. Plánovat podúkol.
	2. Opatřit dostupné standardy rozhraní.
	3. Vybrat standard rozhraní.
	4. Obdržet souhlas Zákazníka.
	5. Příjmout standardy.
	6. Ověřit přijetí standardů.
Krok	***Krok 1. Plánovat podúkol***
Popis:	Podle postupu projektu zahrne Projektový manažer do plánu projektu tyto kroky. Umět si vybrat konkrétní standardy je velmi důležité, protože jejich definování by mohlo být nekonečné, ale když standardy nebudou přijaté, pak bude tato snaha zbytečná. Pokud jde o harmonogram, je žádoucí, aby byly standardy připraveny před zahájením konstrukce komponent s UI. ***2. Opatřit dostupné standardy rozhraní*** - Zjistit, jestli váš Zákazník používá standardy. - Opatřit standardy existujících produktů. Vyhnout se jejich definování od úplného začátku, ve většině projektů je to mimo rozsah, jejich vytvoření může zabrat hodně úsilí a času a vyžaduje schopnosti, které nemusí být dostupné v týmu. ***Krok 3. Vybrat standard rozhraní*** Pokud váš Zákazník nemá standardy, zaúkolujte Programátory, aby vybrali jeden z těch, které byly nalezeny nebo kombinaci z více nalezených. Ohledně dokumentace standardu je nejlehčí cestou implementovat nějaké komponenty a užít je jako prototypy, a pokud máte dostatek času, vytvořte Specifikaci uživatelského rozhraní. V sekci 5 je obsažen odkaz na šablonu uvedenou pod odkazem.

	Krok 4. Obdržet souhlas Zákazníka. Pokud Zákazník nepoptával použitý standard UI, potom můžeme obdržet jeho souhlas prostřednictvím schválení prototypů komponent. Vyhněte se žádosti o schválení standardu samotného, protože to zvětšuje rozsah projektu. Když Zákazník požádá o vytvoření standardu, který nebyl obsažen v počátečních požadavcích, pak by tato změna měla být uveřejněna. ***Krok 5. Příjmout standardy.*** Říct Programátorům, aby přijali standardy v těch komponentách, které mají UI, podporují použití opětovného kódu. ***Krok 6. Ověřit přijetí standardů.*** Ověřit některé komponenty týkající se přijatých standardů UI jinými Programátory.

Podúkol: Definovat standardy pro Konstrukci

Cíl:	Poskytnout vodítko ke kódování softwaru proto, aby byl snadno udržovatelný uvnitř i mimo projekt.
Odůvodnění:	Požadavky na udržitelnost jsou přímo předmětem tohoto podúkolu. Tyto požadavky by měly být dohodnuty v rámci Specifikace požadavků, nicméně ve většině případů nejsou přímo uvedeny, ale jsou vždy očekávány. Chybějící Standardy kódování mohou být vnímány kolegy, kteří se budou snažit upravit kód. Některé komponenty mohou být nahrazeny novými z důvodu neschopnosti jejich pochopení, když je údržba prováděna vývojovým týmem, náklady na projekt se mohou zvýšit, pokud je bude provádět klient sám, budou tyto náklady absorbovány. Standardy kódování by měly být použity alespoň v těch komponentách, které vykonávají klíčovou funkcionalitu. Standardy kódování nejsou přímo uvedeny v 29110-5, nicméně investice trochy úsilí může pomoci zvýšit produktivitu projektu a kvalitu produktu. Užitečné odkazy na konstrukci šablon standardů jsou uvedeny v části 5.
Role:	Projektový manažer
	Technický vedoucí
	Programátor
Artefakty:	Standardy konstrukce
Kroky:	1. Plánovat podúkol.
	2. Opatřit dostupné standardy.

	3. Vybrat standardy.
	4. Přijmout standardy.
	5. Ověřit přijetí standardů.
Krok	***Krok 1. Plánovat podúkol***
Popis:	Podle postupu projektu zahrne Projektový manažer do Plánu projektu tyto kroky. Umět si vybrat konkrétní standardy je velmi důležité, protože jejich definování by mohlo být nekonečné, ale když standardy nebudou přijaté, pak bude tato snaha zbytečná. Co se týká rozvrhu, je žádoucí mít standardy připravené, než začne část Konstrukce, ale samozřejmě to není vždy možné. ***Krok 2. Opatřit dostupné standardy.*** Zjistěte, zda má váš Zákazník Standardy konstrukce, pokud ne snažte se je najít na internetu nebo v jiných dostupných zdrojích. Vyhněte se jejich definování z nástinu, ve většině projektů se dostanete mimo oblast jejich působnosti, vytváření vlastních standardů může stát hodně úsilí a času. ***Krok 3. Vybrat standardy.*** Pokud váš Zákazník nemá standardy, zaúkolujte Programátory, aby vybrali jeden z těch, které byly nalezeny nebo kombinaci z více nalezených. Ohledně dokumentace standardu je nejlehčí cestou implementovat nějaké komponenty a užít je jako příklady, pokud je pro to dostatek času, poté vytvořit Standardy konstrukce. V sekci 5 je obsažen odkaz na šablonu obecného standardu kódování a standardy pro jazyky. ***Krok 4. Přijmout standardy.*** Požadujte, aby Programátoři přijali standardy hned a to hlavně v těch komponentách, které představují klíčovou funkcionalitu. ***Krok 5. Ověřit přijetí standardů.*** Zajistěte, aby u komponent, které provádějí klíčovou funkcionalitu, jiný Programátor ověřil použití standardů.

Činnost: 7.7.7.4 Konstrukce softwaru

Úkoly	Role[5]
SI.4.1 Přidělit úkoly členům pracovního týmu vztahující se k jejich roli dle aktuálního *Plánu projektu.*	Technický vedoucí, Programátor
SI.4.3 Konstruovat nebo aktualizovat *Softwarové komponenty* na základě detailní části *Softwarového návrhu.*	Programátor
SI.4.4 Navrhnout nebo aktualizovat jednotkové testy a použít jich k ověření, že S*oftwarové komponenty* implementují detailní *Softwarový návrh.*	Programátor
SI.4.5 Opravit nalezené chyby až do úspěšného proběhnutí jednotkového testu.	Programátor
SI.4.6 Do Záznamu trasovatelnosti zaznamenat vytvořené či modifikované Softwarové komponenty.	Programátor

Přidělit úkoly členům pracovního týmu

Poznámka: Úkoly týkající se jejich role podle Plánu projektu.

Cíl:	Definujte pořadí Konstrukce a přiřaďte úkoly členům pracovního týmu.
Zdůvodnění:	Většinou identifikovala první verze Plánu projektu většinu komponent, které mají být konstruovány, avšak v této fázi projektu, kdy byly Softwarová architektura a detailní Softwarový návrh kompletní, by měl být zaktualizován Plán projektu, aby obsahoval Konstrukci, v detailu nebo obecně, veškerých komponent, které měly být vyprodukovány. Pořadí konstrukce komponent by mělo být koordinováno s pořadím integrace, aby byly komponenty připravené k integraci v pravý čas. Definování konstrukčního pořadí není obsaženo v 29110-5, avšak investování trocha úsilí k definování pořadí konstrukce může pomoci optimalizovat konstrukční kalendář.
Role:	Technický vedoucí
Produkty:	Plán projektu
Kroky:	1. Opatřit Dokumentaci návrhu softwaru.
	2. Vybrat strategii integrace.
	3. Zdetailizovat Harmonogram projektu.
	4. Přidělit úkoly členům pracovního týmu.

[5] Role jsou definované v další sekci. Role jsou také definované v ISO/IEC 29110-5.1

Krok	***Krok 1. Opatřit Dokumentaci návrhu softwaru***

Popis:	• Opatřete Dokumentaci návrhu softwaru z projektového úložiště, detailní „Low level" Softwarový návrh obsahuje detaily softwarových komponent, které mají být konstruované.
	• Opatřete Záznam trasovatelnosti z úložiště.
	
	Krok 2. Vybrat strategii integrace
	Existuje několik strategií pro určení pořadí integrace, přičemž většina z nich může být rozdělena do neinkrementálních a do inkrementálních, některé z nich jsou:
	Neinkrementální
	- Big bang: integruje najednou všechny části, ve kterých je sestavený a testovaný celý software v jednom kroku
	
	Inkrementální
	- Top-down: Moduly jsou integrovány pohybováním se dolů skrze řídící strukturu.
	- Bottom-up: Moduly nacházející se na nejnižších úrovních jsou integrovány jako první, poté pohybováním se nahoru skrze řídící strukturu.
	
	- Class Test Order: Jde o přímý graf indikující, v jakém pořadí mají být třídy otestované nebo jaké třídy mohou být testovány současně.
	
	Vyberte si Big bang integraci pokud je váš produkt dostatečně malý, aby se s ním dalo zabývat, pokud ne tak Class Test Order udává logický postup shromažďování kódu, můžete také vybrat Top-down nebo Bottom-up podle potřeb vašeho projektu.
	
	Krok 3. Zdetailizovat Harmonogram projektu
	Zdetailizujte Harmonogram projektu obsahující činnosti k rozvoji nebo upravte veškeré identifikované komponenty podle vybrané strategie.
	Dělejte to nejlepší ke zvládnutí časových a finančních nároků projektu, pokud je nutné je upravit, tak by měla být vznesena žádost o změnu řízení. Pro další detaily Harmonogramu projektu získejte informace z Implementačního balíčku Projektového managementu.
	
	Krok 4. Přidělit úkoly členům pracovního týmu
	• Přidělte úkoly členům pracovního týmu
	  ○ Přidělte úkoly pro kódování komponent softwaru
	  ○ Přidělte úkoly pro vývoj testovacích případů a otestujte softwarové komponenty
	• Informujte členy o jejich úkolech

Konstruovat nebo aktualizovat Softwarové komponenty

Poznámka: Komponenty jsou založeny na detailní části Softwarového návrhu.

Cíl:	Vytvořte Softwarové komponenty, tak jak byly navržené.
Odůvodnění:	Programátoři v sobě mohou cítit velikou důvěru k výrobě komponent, aniž by se drželi systematického přístupu, každopádně někteří z nich toto mohou chápat jako velmi užitečné při konstrukci složitých komponent. Existuje mnoho přístupů pro vytváření komponent, zde se používá postup Pseudocode, který je jeden z široko daleko nejvíce akceptovatelných. Jednotlivé kroky byly převzaty z Code complete. (viz odkaz).
Role:	Programátor
Produkty:	Softwarové komponenty
Kroky:	1. Navrhněte komponentu
	2. Naprogramujte komponentu
	3. Ověřte komponentu
Krok Popis:	***Krok 1. Navrhněte komponentu*** **a) Ověřte přispění komponenty.** Podívejte se na Záznam trasovatelnosti a Softwarový návrh pro pochopení přínosu komponenty pro finální produkt. **b) Definujte problém, který bude komponenta řešit.** Uveďte problém, který bude komponenta řešit v dostatečném detailu, aby se povolila její tvorba. Pokud není dostatečný detailní „Low Level" Softwarový návrh, tak by ho měl udělat Programátor za pomoci Návrháře. **c) Zkoumejte funkcionalitu dostupnou v knihovnách standardů.** Zjistěte, zda jsou některé nebo všechny funkce komponenty dostupné v knihovně programovacího jazyka, platformě nebo v nástroji, který používáte. **d) Napište pseudokód.** Začněte s obecným a postupujte k něčemu více specifickému. Nejobecnější částí komponenty je komentář hlavičky, který popisuje co má daná komponenta dělat. Nejprve napište stručné prohlášení o funkci dané komponenty, poté rozložte příkazy do několika nižších úrovní kódu. Vždy si ale pamatujte, že se jedná v zájmu celku. Po vytvoření běžných součástí komponenty následuje další správa věcí, které by mohly škodit v komponentě jako: špatné hodnoty na vstupu, neplatné vstupní hodnoty, neplatné hodnoty vrácené z jiných rutin apod.

e) Navrhněte data komponenty

Definujte klíčové datové typy.

f) Zkontrolujte pseudokód

Překontrolujte pseudokód, pokud si s ním nejste úplně jistí, vysvětlete to ostatním (například navrhovateli softwaru), toto vysvětlení vám pomůže k objasnění vašich nápadů a může daný pseudokód zdokonalit.

Krok 2. Naprogramujte komponentu

Naprogramujte komponenty pomocí užití vybraných programovacích standardů.

a) Napište deklaraci komponenty a proměňte komentář v hlavičce do komentáře v programovacím jazyce.

b) Přeměňte pseudokód do „High-level" komentářů

c) Doplňte kód pod každý komentář
Doplňte kód pod každý řádek pseudokódového komentáře. Každý takovýto komentář popisuje část kódu.

d) Ověřte, jaký kód by měl být dříve zpracován
V nějakých případech se může stát, že budete mít více kódu pod jednou počáteční linií pseudokódu. V tomto případě se zaměřte na tvorbu sub-komponenty.

Krok 3. Ověřte komponentu

Tento krok se skládá z Revize kódu, který napsal Programátor, z kompilace a ladění.

a) Revize kódu

- Zkontrolujte program podle detailního Softwarového návrhu, aby se zajistilo, že jsou implementovány veškeré potřebné funkce programu.
- Následujte Kontrolní seznam Revize kódu, uvedený v sekci 7, aby se našly veškeré programové chyby.
- Při provádění přezkoumávání označte chyby na zdrojovém kódu.
- Postupem revize kódu opravte všechny chyby.
- Po dokončení všech oprav vytvořte zdrojový program uvádějící opravený program.
- Zkontrolujte všechny opravy, aby se zajistilo, že je vše v pořádku.
- Opravte všechny zbylé chyby.

b) Kompilace kódu
Nechte počítač zkontrolovat nedeklarované proměnné, jmenné konflikty atd., toto také udělejte před revizí kódu.

	c) Ladění kódu Jednou, když se zkompiluje, vložte toto do debuggeru a projeďte každý řádek kódu. Přesvědčte se, že každý řádek dělá to, co od něj očekáváte. Toto můžete udělat až po jednotkovém testu a soustředit se na hledání zdroje chyby.

Navrhnout nebo zaktualizovat jednotkové testy a použít je

Cíl:	Najít chyby při kódování jednotkových testů.
Odůvodnění:	Testování komponenty odděleně od zbytku produktu je nejjednodušší cestou k přiřazení chyby komponentě. Chyby, které jsou nalezené při integračních testech, je obtížnější a dražší přiřadit ke správné komponentě. Existuje několik technik, které pomáhají definovat testovací případy. Zde je použita metoda testování na strukturované bázi (Structured Basic Testing). Další techniky jako Data flow, kombinace datových stavů, rozdělení dle ekvivalence, hraniční analýza jsou také doporučované. Následující kroky byly adaptovány z Code complete (viz reference).
Role:	Programátor
Produkty:	Softwarové komponenty [jednotkově testované]
Artefakty:	Testovací případy
Kroky:	1. Definovat počet testovacích případů, které jsou potřeba
	2. Definovat testovací případy
	3. Aplikovat testovací případy
Krok ***Popis***	***Krok 1. Definovat počet testovacích případů, které jsou potřeba*** Myšlenka je taková, že potřebujete otestovat každý příkaz v programu minimálně jednou. Pokud jde o logický příkaz – „if" nebo „while", například – potřebujete rozlišit testování podle toho jak je komplikovaný výraz uvnitř „if" nebo „while" podmínky, abyste se ujistili, že je příkaz plně otestován. Nejjednodušší cestou jak dosáhnout toho, že jste dostali všechny základny, na které se toto vztahuje, je zjistit počet cest skrze celý program a poté vypracovat minimální počet testovacích případů, které vykonají každou cestu skrze program. Můžete spočítat minimální počet případů, které jsou potřebné k základnímu testování v následujícím pořadí. a) Začněte s 1 pro přímočarou cestu skrze proces. b) Přidejte 1 pro každé z následujících klíčových slov nebo jejich ekvivalentu: if, while, repeat, for, and, and or. c) Přidejte 1 pro každý případ v odstavci případů. Pokud odstavec

	případů nemá standardní případ, přidejte 1. ***Krok 2. Definovat testovací případy*** Pro každý případ napište popis podmínek a hodnot proměnných, což pomůže při průchodu programem. Příklad je uveden v druhé části sekce 8. ***Krok 3. Aplikovat testovací případy*** Aplikujte každý testovací případ a zaznamenejte použité hodnoty.

Opravit chyby

Poznámka: Chyby jsou opravované až do úspěšného jednotkového testu (například dosažení konečných kritérií)

Cíl:	Opravit chyby nalezené při jednotkových testech.
Odůvodnění:	Nechat opravit chyby právě nalezené při jednotkovém testu osobou, která je zodpovědná za danou část při tvorbě software, jde o nejrychlejší a nejlevnější cestu, jak chybu opravit.
Role:	Programátor
Produkty:	Softwarové komponenty [opravené]
Kroky:	1. Potvrďit chybu
	2. Opravit chyby
	3. Aplikovat testovací případy (regresní test)
Krok ***Popis:***	***Krok 1. Potvrdit chybu*** Ověřte správnost testovacích případů. ***Krok 2. Opravit chyby*** a) Najděte zdroj chyby. Proveďte krok 3. *Ověřte komponentu* podúkolem „Konstruovat nebo aktualizovat Softwarové komponenty" soustředěně na chybu, aby byla odstraněna. b) Až najdete zdroj chyby, uložte originální zdrojový kód a poté jej změňte. c) Aplikujte odpovídající testovací případ, abyste si ověřili, že je vše vyřešeno. ***Krok 3. Aplikovat testovací případy*** Použijte testovací případy, které jste definovali v úkolu, Aplikujte jednotkové testy, k ověření toho, že opravené části programu nepodsunuli novou chybu do programu.

Aktualizovat Záznam trasovatelnosti

Poznámka: Včetně softwarových komponent konstruovaných nebo modifikovaných.

Cíl:	Zajistit, aby byly veškeré Softwarové komponenty trasovatelné k nejmenší části Softwarového návrhu a aby tyto elementy byly konstruované.
Odůvodnění:	Záznam trasovatelnosti by měl být vytvořen v předchozí fázi projektu, aby se zajistila trasovatelnost od požadavku po elementech návrhu. Tento úkol je věnován k zajištění trasovatelnosti mezi návrhovými a konstrukčními elementy.
Role:	Programátor
Produkty:	Záznam trasovatelnosti [aktualizovaný]
Kroky:	1. Aktualizovat Záznam trasovatelnosti
	Krok 1. Aktualizovat Záznam trasovatelnosti Tento záznam by měl být aktualizován s následujícími informacemi. - Identifikovat softwarové komponenty - Identifikovat testovací případy (nepovinné) - Datum verifikace (např: datum kdy by měla být komponenta otestována a nesmí být zapomenut žádná chyba)

Popis rolí

Zde se nachází abecední výpis rolí, zkratek a výpis kompetencí jako je definováno v ISO 29110 Část 5-1-2.

	Role	***Zkratka***	***Kompetence***
1.	Analytik	AN	Znalost a zkušenost vyvolávající, specifikující a analyzující požadavky. Znalost v navrhování uživatelských rozhraní a ergonomických kritérií. Znalost revizních technik. Znalost editačních technik. Zkušenost na vývoji software a údržbě.
2.	Zákazník	CUS	Znalost zákaznických procesů a schopnosti vysvětlování požadavků Zákazníka. Zákazník musí mít autoritu ke schvalování požadavků a jejich změn. Zákazník zahrnuje zástupce uživatelů, aby zajistil, že bude řešeno provozní prostředí. Znalost a zkušenost v oblasti použití.

3.	Programátor	PR	Znalost a/nebo zkušenost v programování, integraci a jednotkových testech. Znalost revizních technik. Znalost editačních technik. Zkušenost na vývoji softwaru a jeho údržbě.
4.	Technický vedoucí	TL	Znalost a zkušenost v procesu softwarové domény.

Popis produktů

Toto je abecední výpis vstupů, výstupů a interních procesních produktů, jejich popisů, možných stavů a zdrojů produktu.

	Jméno	Popis	Zdroj

1.	Projektový plán	Uvádí, jak budou vykonány projektové procesy a aktivity k zajištění zdárné kompletace a kvality dodávaného produktu. Toto obsahuje následující elementy, které mají podobné charakteristiky, jako následující:
		- *Popis produktu*
		  ○ Účel
		  ○ Obecné požadavky Zákazníka
		- *Popis rozsahu co je součástí a co ne*
		- *Cíle projektu*
		- *Výsledky* - výpis produktů, které mají být doručeny Zákazníkovi
		- *Úkoly*, obsahující verifikaci, validaci a hodnocení se Zákazníkem a pracovním týmem k zajištění kvality pracovních produktů. Úkoly mohou být prezentovány jako struktura rozpisu práce.
		- *Vztahy a dopady úkolů*
		- *Odhadovaná doba trvání úkolů*
		- *Zdroje* (lidé, materiál, vybavení a věci) obsahující požadovaný výcvik a rozvrh, kdy budou zdroje potřebné.
		- *Složení pracovního týmu*
		- *Rozvrh projektových úkolů,* očekávaný začátek a dokončení každého úkolu.
		- *Očekávané úsilí a náklady*
		- *Identifikace rizik projektu*
		- *Verze kontroly strategie*
		  - Identifikovat nástroje projektového úložiště nebo mechanismů
		  - Poloha a přístupový mechanismus pro dané úložiště
		  - Identifikaci verzování a kontroly
		  - Definování mechanismů pro zálohu a obnovu dat
		  - Úložiště, práce s ním a přesuny dat (obsahující archivaci a obnovu)
			Projektový Management

		- *Instrukce pro dodání*
		  - Identifikace elementů pro vydání produktu (např., hardware, software, dokumentace atd.)
		  - Požadavky dodání
		  - Sekvenční řazení úkolů
		  - Identifikace aplikovatelných vydání
		  - Identifikace všech dodaných softwarových komponent s informacemi o verzování
		  - Identifikace potřebných zálohovacích a obnovovacích procedur
		
		Použitelné statusy jsou: verifikovaný, validovaný, změněný a zkontrolovaný.	

2.	Softwarová komponenta	Souhrn souvisejících jednotek kódu. Použitelné statusy jsou: jednotka otestována, opravena a založena.	Softwarová Implementace

3.	Softwarový návrh	Tento dokument obsahuje textové a grafické informace o Softwarové struktuře. Tato struktura může obsahovat následující části:
		Architektonický „High Level" Softwarový návrh – Popisuje kompletní Softwarovou strukturu:
		- Identifikuje potřebné softwarové komponenty
		- Identifikuje vztahy mezi softwarovými komponentami
		- Pozornost je věnována všemu potřebnému:
		  - Charakteristika výkonu software
		  - hardware, software a lidská rozhraní
		  - charakteristika zabezpečení
		  - požadavky návrhu databáze
		  - chování při erroru a obnově dat
		
		Detailizovaný „Low Level" Softwarový návrh – obsahuje detaily softwarových komponent k usnadnění jeho Konstrukce a testů v programovacím prostředí;
		- poskytuje detailní návrh (mohl by být prezentován jako prototyp, diagram, diagram vztahu entit, pseudokód atd.)
		- poskytuje formát vstupních/výstupních dat
		- poskytuje specifikaci potřeb datových úložišť
		- zakládá povinné jmenné konvence
		- definuje formát potřebných datových struktur
		- definuje datová pole a účel jednotlivých datových elementů
		- poskytuje specifikace programových struktur
			Softwarová Implementace

		Použitelné statusy jsou: verifikovaný a znormovaný.	
4.	Záznam trasovatelnosti	Dokumentovat vztahy mezi požadavky obsaženými ve specifikaci požadavků, elementech softwarového návrhu, softwarových komponent, testovacích případů a testovacích procedur. *Toto může obsahovat:* - Identifikuje požadavky trasovatelnosti pro *Specifikaci požadavků* - Poskytuje vpředné a zpětné mapování požadavků pro elementy softwarového návrhu, softwarové komponenty, testovací případy a testovací procedury. Použitelné statusy jsou: verifikovaný, znormovaný a aktualizovaný	Softwarová Implementace

Popis artefaktů

Toto je abecední seznam subjektů, které by mohly být vytvářeny s cílem usnadnit dokumentaci na projektu. Subjekty nejsou vyžadovány v části 5, v které nejsou povinné.

	Jméno	Popis
1.	Standardy konstrukce	Obsahuje konvence, pravidla, idiomy a styly pro: - Zdrojový kód organizace (v závěrce, postupy, třídy, balíčky a jiné struktury) - Dokumentace ke kódu - Moduly a soubory - Proměnné a konstanty - Výrazy - Řídící struktury - Funkce - Zacházení s chybovými stavy – obojí, plánované chyby i výjimky (např.: vstup špatných dat)
2.	Testovací případy	Sada testovacích vstupů, podmínky spuštění a očekávané výsledky vyvinuté pro konkrétní cíle, jako je například jak uplatnit určitý program pro ověření souladu s konkrétním požadavkem [IEEE 1012-2004].
3.	Specifikace uživatelského rozhraní	Tato specifikace zahrnuje: - Okna - Lišta menu - Dialogová okna - Zprávy

	Jméno	Popis
		- Obchodní pravidla mimo jiné

5. Šablona

Odkazy na standardní kódování pomocí šablon

Šablona	**Zdroj**
Kódování Standardní šablona	http://www.construx.com
Zdrojový kód Šablona C / C++	http://www.construx.com
Zdrojový kód Šablona Java	http://www.construx.com
Standardy kódování Java	http://www.ontko.com/java/java_coding_standards.html
Zdrojový kód Šablona Visual Basic	http://www.construx.com
Zdrojový kód Šablona XML	http://www.construx.com

Odkazy ke Specifikaci uživatelského rozhraní pomocí šablon

Šablona	**Zdroj**
Specifikace uživatelského rozhraní	http://www.hcirn.com/tutor/docs/uitempl.php

6. Příklad

Tato sekce pro toto téma obsahuje grafické zobrazení životního cyklu. V tomto příkladu je uvedena pomoc čtenáři, jak implementovat jeho životní cyklus do jeho IT projektu, rámec a omezení.

Bude doplněno.

7. Kontrolní seznam

Kontrolní seznam Revize kódu

Pozn.: Tento kontrolní seznam může být přizpůsoben jazyku, ve kterém programujete.

Značka, která je přidaná na začátku každého bodu kontrolního seznamu, pomáhá k zaznamenání nesrovnalostí při provádění odborné recenze.

Značka - Předmět	**Popis**
CR1 Kompletní	- Ověřit, že všechny funkce v návrhu jsou kódované a že všechny potřebné funkce a postupy byly provedeny.
CR2 Logika	- Zkontrolovat, zda běh programu a všechny procedury a funkce logiky jsou v souladu s detailním návrhem.
CR3 Smyčky	- Provést simulaci všech smyček a rekurzivních procedur, které nebyly nasimulované v detailním návrhu nebo inspekci. - Zajistit, aby každá smyčka byla řádně zahájena a ukončena. - Zkontrolovat, že každý cyklus je vykonán správným počtem opakování.
CR4 Volání procedur	- Zkontrolovat, zda všechny funkce a volání procedury, přesně odpovídají definici formátů a typů.
CR5 Deklarace	Ověřit, že všechny proměnné a parametry: - Jsou pouze jednou deklarované - Používají se pouze v rámci svého nadeklarované rozsahu - Jsou korektně použity kdekoliv v kódu
CR6 Inicializace	- Zkontrolovat, že všechny proměnné jsou inicializované
CR7 Limity	- Zkontrolovat všechny proměnné, pole a indexy, že jejich použití nebude překračovat deklarované limity
CR8 Začátek-konec	- Zkontrolovat všechny *„begin-end"* dvojice nebo jejich ekvivalenty, obsahující případy, v kterých mohou být chybně použité.
CR9 Boolean	- Zkontrolovat logické podmínky.
CR10 Formát	- Zkontrolovat každý řádek programu určený k výuce na formát a interpunkci.
CR11 Ukazatele	- Zkontrolovat, že všechny ukazatele jsou správně použity.
CR12 Vstup-výstup	- Zkontrolovat všechny vstupně-výstupní formáty.
CR13 Speling	- Zkontrolovat, že každá proměnná, parametr a klíčová práce je pravopisně správně.
CR14 Comments	- Zajistit, aby veškeré komentáře, splňovali normu.

8. Nástroj

Matice trasovatelnosti

- Cíle:
  - Udržet spojení od zdroje každého požadavku přes jeho rozklad k implementaci a testování (ověřování).
  - Zajistit, aby všechny požadavky byly určeny a jen to co bylo požadováno bylo vyvinuto.
  - Užitečné při provádění posuzování dopadů na požadavky, návrh nebo dalších měnitelných položek

Poznámka: Matice trasovatelnosti je rozšířený a používaný nástroj k realizaci Záznamu trasovatelnosti.

Matice trasovatelnosti

Matice sledovatelnosti

Datum (rr-mm-dd): _____

Název projektu:_____

	Název (Tisk)	**Podpis**	**Datum (rr-mm-dd)**
Verifikováno:	_____	_____	_____
Schváleno:	_____	_____	_____

Identifikační číslo	Potřebný text	Text požadavku	Verifikační metoda	Název nebo ID modelového případu	Název nebo ID modulu kódování	Název nebo ID testovací procedury	Datum verifikace	Jméno osoby, která provedla ověření	Výsledek ověření

Instrukce	
Výše uvedená tabulka by měla být vytvořena v excelovském dokumentu nebo databázi, tak aby šla jednodušše setřídit dle každého z jednotlivých sloupců, a také aby šlo jednoduše oboustranného trasování. Jednoznačné identifikátory pro položky by měly být přiřazeny v hierarchické formě osnovy tak, aby na nejnižší úrovni (tj. detailnější) položek šlo sledovat změny až do vyšších položek.	
Unikátní identifikace požadavku (ID)	Unikátní identifikace požadavku / Prohlášení o Systémových požadavcích, kde je odkazováno na požadavek, a / nebo jednoznačnou identifikaci (ID) pro rozložené požadavky
Popis požadavku	Zadejte popis požadavku (např. žádosti o změnu popisu).
Návrh Reference	Zadejte číslo odstavce, kde je odkazováno na CR v projektové dokumentaci
Modul / Nakonfigurované referenční položky	Zadejte jedinečný identifikátor softwarového modulu, nebo nastavenou položku, kde je realizován návrh.

Uvolněné reference	Zadejte verzi / číslo buildu verze, kde je splněn požadavek
Název testovacího skriptu/Krok pod referenčním číslem	Zadejte testovací skript jméno / číslo kroku, kde je požadavek odkazuje (např. krok 1)

Cíle

Trasovatelnost požadavků by měla:

- Zajistit trasovatelnost pro každou úroveň projektu. Zejména:
  - Zajistit, že každý nižší požadavek lze vysledovat na vyšší úrovni nebo z originálního zdroje
  - Zajistit, že každý návrh, implementace a testování prvků lze vysledovat na základě požadavku
  - Zajistit, že každý požadavek je zastoupen v návrhu a implementaci
  - Zajistit, že každý požadavek je zastoupen při testování / ověření
- Zajistit, že trasovatelnost změn je používána při hodnocení požadavků ve změnách Plánu projektu, činnostech, produktech
- Být udržována a aktualizována jakmile nastanou nějaké změny.
- Být konzultován při přípravě posuzování dopadů pro každý návrh na změnu projektu.
- Aby bylo počítáno s tím, že zachování je časově náročný a pracný proces, který by měl být sledován a tomuto sledování by měl být přiřazen nějaký člen týmu
- Být zachována jako elektronický dokument

Příklad vytvoření testovacích případů s použitím metody testování na strukturované bázi (Structured Basis testing) pro programy v Javě, výtah z knihy Test Complete.

```
1  // Compute Net Pay       <-- 1
2  totalWithholdings = 0;
3
4  for ( id = 0; id < numEmployees; id++ ) {       <-- 2
5
6     // compute social security withholding, if below the maximum
7     if ( m_employee[ id ].governmentRetirementWithheld < MAX_GOVT_RETIREMENT ) {       <-- 3
8          governmentRetirement = ComputeGovernmentRetirement( m_employee[ id ] );
9     }

10
11    // set default to no retirement contrDPution
12    companyRetirement = 0;
13
14    // determine discretionary employee retirement contrDPution
15    if ( m_employee[ id ].WantsRetirement &&       <-- 4
16       EligDPleForRetirement( m_employee[ id ] ) ) {
17       companyRetirement = GetRetirement( m_employee[ id ] );
18    }
19
20    grossPay = ComputeGrossPay ( m_employee[ id ] );
21
22    // determine IRA contrDPution
23    personalRetirement = 0;
24    if ( EligDPleForPersonalRetirement( m_employee[ id ] ) ) {       <-- 5
25       personalRetirement = PersonalRetirementContrDPution( m_employee[ id ],
26          companyRetirement, grossPay );
27    }
28
29    // make weekly paycheck
30    withholding = ComputeWithholding( m_employee[ id ] );
```

```
31    netPay = grossPay - withholding - companyRetirement - governmentRetirement -
32       personalRetirement;
33    PayEmployee( m_employee[ id ], netPay );
34
35    // add this employee's paycheck to total for accounting
36    totalWithholdings = totalWithholdings + withholding;
37    totalGovernmentRetirement = totalGovernmentRetirement + governmentRetirement;
38    totalRetirement = totalRetirement + companyRetirement;
39 }
40
41 SavePayRecords( totalWithholdings, totalGovernmentRetirement, totalRetirement );
```

(1)Count "1" for the routine itself.

(2)Count "2" for the for.

(3)Count "3" for the if.

(4)Count "4" for the if and "5" for the &&.

(5)Count "6" for the if.

V tomto případě budete potřebovat jednu počáteční zkoušku a ještě jednu pro každou z pěti klíčových slov, tedy celkem šest. To neznamená, že šest zkušebních případů se bude týkat všech základen. To znamená, že na minimum, je třeba šest případů. Pokud jsou případy konstruovány pečlivě, téměř určitě se nebudou vztahovat na všechny základny. Trik je v tom dávat pozor na stejná klíčová slova, které jste použili při počítání množství v nutných případech. Každé klíčové slovo v kódu představuje něco, co může být buď pravdivé nebo nepravdivé, ujistěte se, že máte alespoň jeden modelový případ pro každou pravdu a alespoň jeden pro každou nepravdu.

Zde je soubor testovacích případů, který zahrnuje všechny základny v tomto příkladu:

Případ	**Popis testu**	**Testovací data**
1	Nominální případ	All boolean conditions are true
2	Počáteční podmínka *for* je *false*	numEmployees < 1
3	Počáteční podmínka *if* je *false*	m_employee[id].governmentRetirementWith-held >=MAX_GOVT_RETIREMENT
4	Druhá podmínka *if* je *false* protože její první část *and* je *false*	not m_employee[id].WantsRetirement
5	Druhá podmínka *if* je *false* protože její druhá část *and* je *false*	not EligDPleForRetirement (m_employee[id])
6	Třetí podmínka *if* je *false*	not EligDPleForPersonalRetirement (m_employee[id])

9. Reference k jiným standardům a modelům

Tato část obsahuje odkazy na zavedení balíčku na vybrané ISO a ISO / IEC a Capability Maturity Model IntegrationSM verze 1.2 Software Engineering Institute (CMMI ®).

Poznámky:

- Tato část je poskytována pouze pro informační účely.
- Pouze úkoly, na které se vztahuje tento balíček, jsou pro nasazení uvedeny v každé tabulce.
- Tabulky používají následující konvence
    - Plné pokrytí = F
    - Částečné pokrytí = P
    - Žádné pokrytí = N

ISO 9001 Matice referencí

Název úkolu a krok	Pokrytí F/P/N	Klauzule ISO 9001	Komentáře
SI.2.2 Dokumentovat nebo aktualizovat Specifikaci požadavků.	P	7.3.2 Vstupy návrhu a vývoje d) ostatní požadavky nezbytné pro návrh a vývoj	
Podúkol: Vybrat standard uživatelského rozhraní.			Mohli by být specifikovány Zákazníkem, v takovém případě jsou uvedeny v 7.2.1
Podúkol: Definovat standardy pro Konstrukci.			Většina případů, které nejsou uvedeny Zákazníky, ale jsou důležité pro některé součásti.
SI.4.1 Přidělit úkoly členům pracovního týmu vztahující se k jejich roli dle aktuálního *Plánu projektu.*	P	7.3.1 Plánování návrhu a vývoje	Obsahuje pouze komunikační úlohy.
SI.4.3 Konstruovat nebo aktualizovat *Softwarové komponenty* na základě detailní části *Softwarového návrhu.*	P	7.3.3 Výstupy návrhu a vývoje a) splňují vstupní požadavky na návrh a vývoj.	

SI.4.4 Navrhnout nebo aktualizovat jednotkové testy a použít jich k ověření, že S*oftwarové komponenty* implementují detailní *Softwarový návrh.*	P	7.3.4 Návrh a vývoj hodnocení a) vyhodnotit schopnost výsledků návrhu a vývoje pro splnění požadavků, a b) identifikovat problémy a navrhnout potřebná opatření.	
SI.4.5 Opravit nalezené chyby až do úspěšného proběhnutí jednotkového testu.	P	7.3.4 Hodnocení návrhu a vývoje a) vyhodnotit schopnost výsledků návrhu a vývoje pro splnění požadavků, a b) identifikovat problémy a navrhnout potřebná opatření.	
SI.4.6 Do Záznamu trasovatelnosti zaznamenat vytvořené či modifikované Softwarové komponenty.	P	7.3.7 Řízení navrhovaných a vývojových změn.	

ISO/IEC 12207 Matice referencí

Název úkolu a krok	**Pokrytí F/P/N**	**Klauzule ISO/IEC 12207**	**Komentáře**
SI.2.2 Dokumentovat nebo aktualizovat Specifikaci požadavků.	P	7.1.2.3.1 Analýza softwarových požadavků. 7.1.2.3.1.1 Implementátor musí vytvořit a dokumentovat požadavky na software (včetně kvality charakteristiky specifikace) které jsou popsány níže. b) Rozhraní na externí softwarové položky. k) požadavky na jeho údržbě.	a) Vztahuje se pouze na požadavky, které mají vliv na fázi Konstrukce. b) je spojen s UI, a k) s konstrukčními normami.
Podúkol: Vybrat standard uživatelského rozhraní.		7.1.4.3.1 Detailní Softwarový návrh. 7.1.4.3.1.2 Implementátor musí stanovit a zdokumentovat detailní návrh pro rozhraní externí software položky, mezi softwarové komponenty a software mezi jednotkami. Podrobný návrh rozhraní povolí kódování bez potřeby dalších informací.	Standard rozhraní je užitečné při provádění 7.1.4.3.1.2

Podúkol: Definovat standardy pro Konstrukci.		7.1.5.3.1 Konstrukce softwaru. 7.1.5.3.1.5 Implementátor vyhodnotí softwarový kód a výsledky testů na základě kritérií uvedených níže. Výsledky hodnocení musí být dokumentovány. e) Vhodnost metod kódování a norem.	Konstrukční normy budou užitečné při provádění 7.1.5.3.1.5
SI.4.1 Přidělit úkoly členům pracovního týmu vztahující se k jejich roli dle aktuálního *Plánu projektu.*	P	6.3.1.3.3 Projekt aktivace. 6.3.1.3.3.3 manažer zahájí implementaci a nastaví kritéria, které vykonávají kontrolu nad projektem.	Tento úkol se aktivuje pouze ve fázi výstavby.
SI.4.3 Konstruovat nebo aktualizovat *Softwarové komponenty* na základě detailní části *Softwarového návrhu.*	P	7.1.5.3.1 Konstrukce softwaru. 7.1.5.3.1.1 Implementátor musí stanovit a zdokumentovat následující: a) Každou jednotku software a databazi.	
SI.4.4 Navrhnout nebo aktualizovat jednotkové testy a použít jich k ověření, že S*oftwarové komponenty* implementují detailní *Softwarový návrh.*	P	7.1.5.3.1 Konstrukce softwaru. 7.1.5.3.1.1 Implementátor musí stanovit a zdokumentovat následující: b) Zkušební postupy a data pro testování jednotlivých jednotek a softwaru databáze.	
SI.4.5 Opravit nalezené chyby až do úspěšného proběhnutí jednotkového testu.	P	7.1.5.3.1 Konstrukce softwaru. 7.1.5.3.1.2 Implementátor testuje každé softwarové zařízení a databáze zajišťuje, že jsou splněny požadavky. Výsledky testů musí být zdokumentovány.	
SI.4.6 Do Záznamu trasovatelnosti zaznamenat vytvořené či modifikované Softwarové komponenty.	P	7.1.5.3.1 Konstrukce softwaru. 7.1.5.3.1.5 Implementátor evaluuje kód softwaru a výsledky testování musí být dokumentovány. a) návaznost na požadavky a návrh software položky.	

CMMI matice referencí

Název úkolu a krok	Pokrytí F/P/N	PA/Cíl/ Postup CMMI V1.2	Komentáře
SI.2.2 Dokumentovat nebo aktualizovat Specifikaci požadavků.	P	Realizace požadavků (RD) SG-1 Realizace požadavků Zákazníka SP 1.2 Realizace požadavků Zákazníka	Omezení pro ověřování a schvalování nejsou zahrnuty.
Podúkol: Vybrat standard uživatelského rozhraní.			Poskytnout zúčastněným stranám ve fázi konstrukce
Podúkol: Definovat standardy pro Konstrukci.			Poskytnout zúčastněným stranám ve fázi konstrukce
SI.4.1 Přidělit úkoly členům pracovního týmu vztahující se k jejich roli dle aktuálního *Plánu projektu.*	P	Technické řešení (TS) SG 3 Implementovat Návrh projektu GP 2.8 Sledovat a řídit procesy	Aktivuje pouze plán realizace.
SI.4.3 Konstruovat nebo aktualizovat *Softwarové komponenty* na základě detailní části *Softwarového návrhu.*	P	Technické řešení (TS) SG 3 Implementovat Návrh produktu SP 3.1 Implementace návrhu	Týká se praxe 1. Užívat účinné metody implementace produktových komponent 2. Dodržujte platné standardy a kritéria.
SI.4.4 Navrhnout nebo aktualizovat jednotkové testy a použít jich k ověření, že S*oftwarové komponenty* implementují detailní *Softwarový návrh.*	P	Technické řešení (TS) SG 3 Implementovat Návrh produktu SP 3.1 Implementovat Návrh	Týká se praxe 4. Proveďte jednotkové testy výrobku, komponent podle potřeby.
SI.4.5 Opravit nalezené chyby až do úspěšného proběhnutí jednotkového testu.	P	Technické řešení (TS) SG 3 Implementovat Návrh produktu SP 3.1 Implementovat Návrh	Týká se praxe 4. Proveďte jednotkové testy komponent produktu podle potřeby.
SI.4.6 Do Záznamu trasovatelnosti zaznamenat vytvořené či modifikované Softwarové komponenty.	P	Řízení požadavků (REQM) SG 1 Spravovat požadavky SP 1.4 Udržovat obousměrné trasovatelnosti požadavků.	To se vztahuje pouze na trasovatelnost konstruovaných komponent.

10. Reference

Klíč	Reference
[Code Complete]	Steve McConnell, Code Complete, Second Edition, Redmond, Washington, Microsoft Press, 2004.
[PMBOK]	A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Fourth Edition, Project Management Institute, 2008
[IEEE 1012-2004]	IEEE 1012-2004 IEEE Standard for Software Verification and Validation, IEEE Computer Society
[ISO/IEC 12207]	ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes.
[ISO/IEC 29110-5-1-2]	ISO/IEC TR 29110-5-1-2:2011, Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs) – Part 5-1-2: Management and Engineering Guide – Basic VSE Profile
[ISO/IEC 24765]	ISO/IEC 24765:2011 Systems and software engineering vocabulary
[ISO/IEC 20926]	ISO/IEC 20926:2003 Software engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting practices manual
[ISO/IEC 29881:2008]	ISO/IEC 29881:2008 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method,
[IEEE 1233-1998]	IEEE Guide for Developing System Requirements Specifications

11. Hodnotící formulář:

Implementační balíček, Konstrukce a Jednotkové testování verze 0.4

Vaše zpětná vazba nám umožní zlepšit tento implementační balíček, jsou vítány Vaše připomínky a návrhy.

1. Jak jste spokojeni s obsahem tohoto implementačního balíčku?

☐ *Velmi spokojeni* ☐ *Spokojeni* ☐ *Ani spokojen ani nespokojen* ☐ *Nespokojený* ☐ *Velmi nespokojený*

2. Pořadí, ve kterém jsou témata probírána, jsou logická a snadno sledovatelná?

☐ *Velmi spokojeni* ☐ *Spokojeni* ☐ *Ani spokojen ani nespokojen* ☐ *Nespokojený* ☐ *Velmi nespokojený*

4. Jak jste byli spokojeni s výskytem / formátem tohoto implementačního balíčku?

☐ *Velmi spokojeni* ☐ *Spokojeni* ☐ *Ani spokojen ani nespokojen* ☐ *Nespokojený* ☐ *Velmi nespokojený*

4. Byla zde nějaká zbytečná témata?

5. Jaké chybějící byste rádi viděli v tomto balíčku? (Prosím popiš)

- Navrhované téma
- Důvody pro nová témata

6. Nějaká chyba v balíčku?

- Uveďte prosím:
  - Popis chyby:
  - Místo nalezeních chyby (část #, obrázek #, tabulka #) :

7. Jiná zpětná vazba nebo komentář:

8. Doporučil byste tento implementační balíček kolegovi z jiného velmi malého podniku?

☐ *Rozhodně* ☐ *Pravděpodobně* ☐ *Nejsem si jist* ☐ *Pravděpodobně ne* ☐ *Rozhodně ne*

Dobrovolné

- Jméno: _____
- Emailová adresa : _____

Emailové adresy, na které můžete poslat tento formulář: claude.y.laporte@etsmtl.ca or Avumex2003@yahoo.com.mx

12. Souhrn pojmů převedených do českého jazyka

Deployment package	Implementační balíček
Sub-task	Podúkol
Construction	Konstrukce
Template	Šablona
Task	Úkol
Role	Role
Project Plan	Projektový plán
Software parts	Softwarové složky
Software design	Softwarový návrh
Project schedule	Harmonogram projektu
Traceability record	Záznam trasovatelnosti
Component	Komponenta

13. Komentář k implementačnímu balíčku

Český

Implementační balíček Konstrukce a Jednotkové testování se zabývá Konstrukcí a Jednotkovým testováním. Jde o jeden z několika balíčků, který se zařazuje do kompletu Basic profil, jenž je definován v ISO/IEC 29110 v části 5-1-2: manažerská a technická příručka. Konkrétně popisuje, jaké kroky mají následovat během konstrukce software a jaké kroky mají následovat během jednotkového testu.

Po lokalizaci implementačního balíčku a jeho lepším pochopení se dá říci, že daný balíček je schopný usnadnit práci a je použitelný samostatně bez implementace dalších definovaných procesů v normě ISO/IEC 29110. Dá se říci, že tvoří použitelný návod s příklady, který je poměrně snadno pochopitelný a použitelný. Lze si tedy představit, že podle něj budou realizované skutečné projekty.

Po porovnání implementačního balíčku Konstrukce a Jednotkové testování s normou ISO/IEC 29110 část 5-1 Engineering and Management Guide – Basic VSE Profile jsme dospěli k závěru, že obsah balíčku z větší části odpovídá normě, ale některé úkoly se v implementačním balíčku neobjevují vůbec. Například v činnosti 7.7.7.4. Softwarová konstrukce je zde uveden jako poslední krok číslo 4.6, ale v normě končí činnost krokem 4.7. Další nedostatek implementačního balíčku je v úvodu, kde jsou uvedeny konkrétní a obecné termíny použité v implementačním balíčku. Zde je řečeno, že specifické termíny se vyskytují pouze v tomto balíčku, nicméně při průzkumu dalších balíčků bylo zjištěno, že tyto „specifické" termíny se vyskytují i v jiných balíčkách. Činnost 7.7.7.2 Analýza softwarových požadavků v tomto balíčku se může zdát jako zbytečná, protože existuje jiný implementační balíček, který je specializovaný přímo na tuto oblast – Analýza požadavků (Requirements analysis), nicméně není od věci se zde zmínit o požadavcích, které jsou specifické pro tento implementační balíček. Struktuře implementačního balíčku nelze nic vytknout, veškeré úkoly, role k nim apod. jsou jasně strukturované. Drobným nedostatkem je automaticky se aktualizující pole s datem poslední aktualizace uvedené na první stránce implementačního balíčku. Může to svádět k myšlence, že byl bylíček přepracován, ale nemuselo se tak stát.

Vzhledem k tomu, že tento balíček je určen především pro začínající organizace, které nemají dostatečnou zkušenost s vývojem softwaru, je lepší pro úplné pochopení smyslu a účelu implementačního balíčku ho několikrát pečlivě pročíst a vytvářet si tak jasnou představu o tom, co se odehrává, vzniká apod. Tato skutečnost vyplývá z vlastní zkušenosti.

Anglický

The implementation package Construction and unit testing is dealing with construction (of what) and unit test. It is one of the packages arranged in set that is called Basic profile, which is defined in ISO/IEC 29110 standard in part 5-1-2: manager and technical/technician guide. It describes specifically which steps should follow in the construction process and in the unit test.

After localization of the implementation package and its better understanding one can say, that this package is able to facilitate the work and can be used independently without any implementation of other processes defined in the ISO/IEC 29110 standard. We can say it is useable guide with examples, which one can easily understand to and use it. One can imagine that *pro futuro* [to future – lat.] it will be used for realization of actual projects.

After comparison of the implementation package Construction and unit testing with the ISO/IEC 29110 standard, part 5-1: Engineering and Management Guide - Basic VSE Profile, we concluded that the most part of package's content correspondents to the standard, but there are some tasks that do not appear in implementation package at all. For example activity 7.7.7.4. Software construction describes step 4.6 as the last one, but the standard describes also step 4.7. Another flaw of the package is in its introduction, where are specified specific and general terms used in this package. It says that some of the specific terms occur only in this package, but we found them also in some other different packages. Activity 7.7.7.2. Software requirements analysis in this package may seem to be unnecessary because there is another implementation package that is specialized exactly on this area - Requirements analysis. It is appropriate to mention the requirements specific for this implementation package. The structure of the package is almost beyond reproach; all tasks, roles etc. are clearly structured. Minor drawback is in automatically updating field which contains date of last update (actualization). One could thus think the package was revised, which would not have to be truth.Given that this package is designed especially for new organizations that do not have sufficient experience in software development, for its full understanding of the meaning and purpose of the implementation package it is better to read it carefully few times and thus get a clear idea of what exactly is happening, arising etc. This fact is based on our own experience.