Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	ZS 20/21
Autoři	Anna Vágnerová, vaga00 Veronika Süssenbeck, polv03 Zdeněk Tomka, tomz03
Téma	Metody měření funkční velikosti softwaru
Datum odevzdání	17.12.2020

Abstrakt:

Tato práce se zabývá metodami pro měření funkční velikosti softwaru a jako příklad jsou popsány metody IFPUG FPA a COSMIC-FFP. Tyto metody jsou analyzovány a následně porovnány. V práci se nachází představení společných znaků a rozdílů v kategoriích technické aspekty, vnímání data jejich pohybu a dostupnosti. Výsledky této práce umožňují vybrat čtenáři vhodnou metodu pro měření funkční velikosti softwaru na základě jeho potřeb.

Klíčová slova: měření funkční velikosti softwaru, IFPUG, COSMIC, funkční body

Obsah

Úvod

Měření funkční velikosti softwaru je důležité pro odhadování nákladů jak pro dodavatele softwaru, tak pro zákazníka. Kvalitní odhady ať už finanční, časové, zdrojové, rozsahu či pracnosti jsou nedílnou součástí každého úspěšně vedeného projektu. Proto jsou postupně vyvíjeny metody odhadů, které společnostem pomáhají doručovat projekty úspěšně v definovaných mezích podpořených přesnými estimacemi.

Tato semestrální práce se bude zabývat popisem a srovnáním metod pro měření funkční velikost softwaru. Protože se v rámci výkladu metod měření funkční velikosti setkáváme s pojmem požadavek, bude tomuto pojmu věnována kapitola Požadavky na software. Následně jsou v rámci samostatné kapitoly stručně sepsány existující ISO normy, které se právě měřením funkční velikosti softwaru zabývají. V další kapitole se bude práce věnovat standardu IFPUG a ISO normě 20926, která ji pokrývá, konkrétně její nejnovější verzi z roku 2009. V rámci této kapitoly bude popsán postup výpočtu funkčních bodů. V další kapitole bude popsána metodika COSMIC-FFP a shodně s předchozí kapitolou také související ISO norma, v tomto případě 19761 z roku 2011. Závěr práce se bude zabývat srovnáním těchto dvou metod, jejich společných a rozdílných znaků. Tato práce se nezabývá podrobným rozborem ostatních ISO norem a souvisejících metod představených v kapitole Standardy pro měření funkční velikosti softwaru.

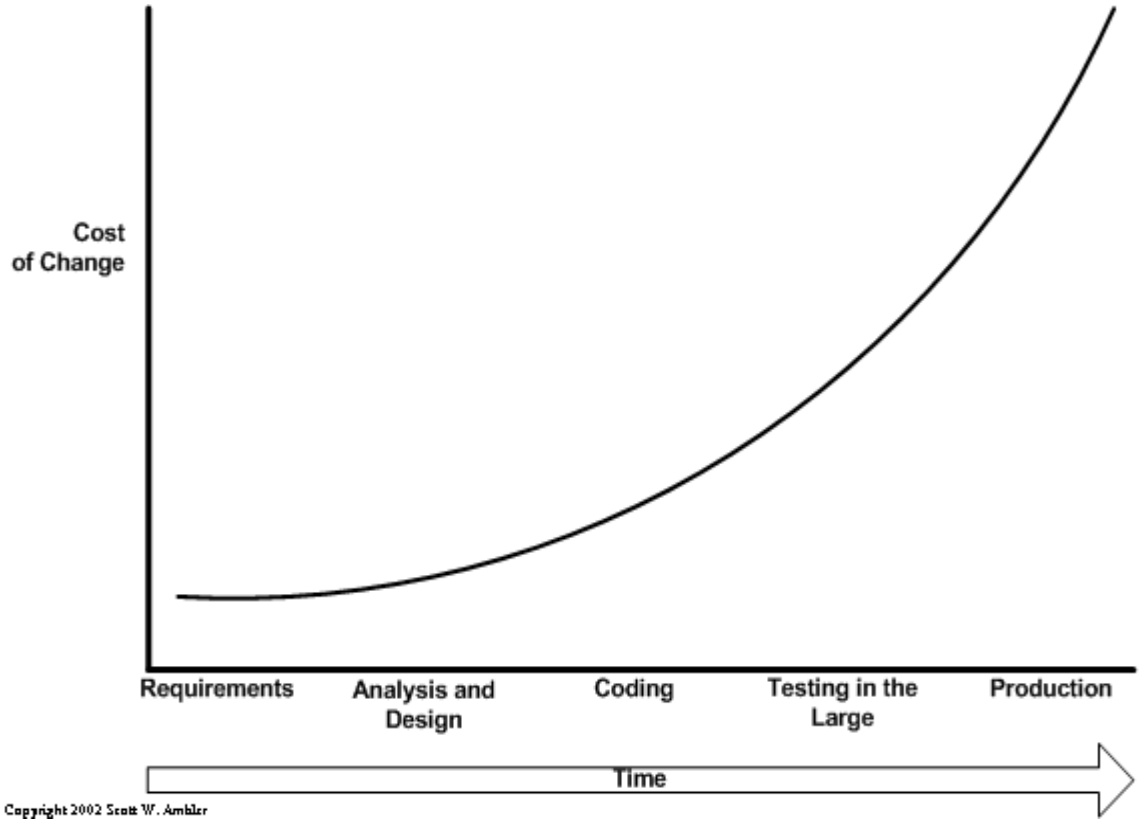
Hlavním cílem této práce je srovnání metod měření funkční velikosti softwaru, konkrétně metod IFPUG FPA a COSMIC FFP. Dílčí cíle, díky kterým bude dosaženo cíle hlavního, je představit metody pro měření funkční velikosti softwaru na základě uživatelských požadavků IFPUG a COSMIC a standardy ISO, které tyto metody pokrývají.

Při psaní této semestrální práce budou použity následující metody, které dopomohou k naplnění cílů: Analýza a následná syntéza dostupných literárních zdrojů, odborných článků, již napsaných závěrečných prací a vlastních poznatků.

Na tuto práci lze navázat porovnáním všech standardů zabývajících se funkční velikosti softwaru.

1. Požadavky na software

Různé studie naznačují, že chyby vzniklé během činností souvisejících s požadavky představují 40 až 50 procent všech závad zjištěných u softwarového produktu. (Davis, 2005) Analýza požadavků je jednou z počátečních fází vývoje Software. Správná definice požadavků má velký vliv na úspěšné dodání produktu v rámci rozpočtu a času. Obrázek 1 níže graficky znázorňuje závislost času (postupu v životním cyklu) a nákladů na změnu. S každou následující fází práce na produktu se exponenciálně zvyšuje cena za provedení změny produktu. Je tedy velice důležité provádět analýzu požadavků důsledně.



Obrázek 1: Náklady na změnu v průběhu vývoje softwaru (AMBLER, 2020)

Co se však rozumí pod pojmem *požadavek* v kontextu softwarového inženýrství? Představa jedinců o tom, co je požadavek se může různit a setkáme se různými pojmy jako například uživatelský požadavek, softwarový požadavek, byznys požadavek, funkce, omezení atd. Definice požadavků zákazníkem může vývojářovi znít jako povrchový koncept produktu, zatímco vývojářova definice požadavku může uživateli připadat jako podrobný návrh uživatelského rozhraní. (Weigers, 2013) Proto si níže uvedeme, jak požadavek definují různé zdroje.

Požadavky jsou specifikací toho, co by mělo být implementováno. Jsou to popisy toho, jak by se měl systém chovat, nebo vlastnosti či atributy systému. Mohou být omezením procesu vývoje systému. (Sawyer, 1997)

Standard Glossary of Software Engineering Terminology (IEEE, 1990) definuje požadavek následovně:

1. *Podmínka nebo schopnost potřebná uživatelem k vyřešení problému nebo dosažení cíle.*
2. *Podmínka nebo schopnost, kterou musí splňovat nebo mít systém nebo systémová složka, aby bylo naplněna smlouva, norma, specifikace nebo jiný formálně uložený dokument.*
3. *Dokumentované znázornění podmínky nebo schopnosti jako v bodě 1 nebo 2.*

Cokoliv, co řídí možnosti návrhu (Lawrence, 1997)

1.1. Úrovně a typy požadavků

Protože existuje tolik různých typů požadavků, potřebujeme konzistentní soubor přídavných jmen, abychom mohli porozumět přetíženému termínu „požadavek“. Tabulka níže představuje definice, které použijeme pro některé termíny běžně se vyskytující v oblasti požadavků.

Tabulka 1: Typy informací předávaných požadavky (Wiegers, 2013)

Termín	Definice
Byznys požadavek	Obchodní cíl organizace, která produkt vytváří, nebo zákazníka, který jej pořizuje.
Byznys pravidlo	Politika, pokyny, norma nebo regulace, která definuje nebo omezuje určitý aspekt podnikání. Ne požadavek na software jako takový, ale původ několika typů softwarových požadavků.
Omezení	Omezení možností, které má vývojář k dispozici při návrhu a implementaci produktu.
Požadavek na externí rozhraní	Popis spojení mezi softwarovým systémem a uživatelem, jiným softwarovým systémem nebo hardwarovým zařízením.
Funkce	Jedna nebo více logicky souvisejících systémových schopností, které poskytují hodnotu uživateli a jsou popsány souborem funkčních požadavků.
Funkční požadavek	Popis chování, které systém bude vykazovat za určitých podmínek.
Nefunkční požadavek	Popis vlastnosti nebo charakteristiky, kterou musí systém vykazovat, nebo omezení, které musí respektovat.
Atribut kvality	Druh nefunkčního požadavku, který popisuje službu nebo výkonnostní charakteristiku produktu.
Systémový požadavek	Požadavek nejvyšší úrovně pro produkt, který obsahuje více subsystémů, což může být veškerý software nebo software a hardware.
Uživatelský požadavek	Cíl nebo úkol, který mohou určité skupiny uživatelů provádět se systémem, nebo požadovaný atribut produktu.

1.2. Ideální charakteristiky požadavku

Podle knihy *Software requirements* (Wiegers, 2013) by v ideálním světě každý jednotlivý byznys, uživatelský, funkční i nefunkční požadavek vykazoval vlastnosti popsané níže.

Kompletní

Každý požadavek musí obsahovat všechny informace nezbytné k tomu, aby mu čtenář porozuměl. V případě funkčních požadavků to znamená poskytnutí informací, které vývojář potřebuje, aby byl schopen je správně implementovat. Pokud je známé, že určité informace chybí, je potřeba doplnit k popisu požadavku TBD (bude určeno).

Správný

Každý požadavek musí přesně popisovat schopnost, která bude vyhovovat potřebám nějakého zainteresovaného subjektu a musí jasně popisovat funkčnost, která má být vybudována. Požadavky nesmí být ve vzájemném konfliktu. Pro posouzení správnosti požadavku je potřeba, aby byl požadavek zkontrolován zástupci uživatelů.

Proveditelný

Každý požadavek musí být možné realizovat v rámci schopností a omezení systému a dále v rámci omezení projektových tedy časových, rozpočtových a personálních. Vývojář, který se účastní sběru požadavků může poskytnout kontrolu, zdali je požadavek technicky proveditelný a co je možné udělat s užitím nadměrných nákladů či úsilí. Proveditelnost lze také vyhodnotit přístupem přírůstkového vývoje nebo prototypováním.

Nezbytný

Každý požadavek by měl popisovat schopnost, která zúčastněným stranám poskytuje předpokládanou obchodní hodnotu, odlišuje produkt na trhu nebo je vyžadována pro splnění vnějšího standardu, politiky nebo regulace. Každý požadavek byste měli být schopni vztáhnout k obchodnímu cíli, který jasně indikuje, proč je nezbytný.

Prioritizovaný

Požadavky by měli být upřednostňovány podle toho, které jsou nejdůležitější pro dosažení požadované hodnoty. Přiřazením priority požadavku určíme, jak důležitá je jeho implementace v rámci konkrétního *release* (vydání) produktu. Stanovení priorit požadavků by mělo být spoluprací zahrnující více hledisek zúčastněných stran.

Jednoznačný

Jak již název napovídá, jedná se o vlastnost, kdy je požadavek jednoznačný pro všechny zúčastněné v procesu tvorby produktu. Toho lze dosáhnout například formálním vzájemným hodnocením, kdy jednotlivci mezi sebou porovnávají jejich chápaní požadavku.

Přirozený jazyk je náchylný ke dvěma typům nejednoznačnosti. Jeden typ pozná každý sám, a to tehdy, když jej napadne více způsobů, jak interpretovat daný požadavek. Druhý typ nejasností je těžší zachytit. Konkrétně, když si různí lidé požadavek přečtou a přijdou s různými výklady. Požadavek dává smysl každému z nich, ale pro každého z nich znamená něco jiného.

Ověřitelný

Tato vlastnost určuje, zdali tester bude schopen ověřit, že je požadavek správně implementován. Neověřitelné jsou například požadavky, které jsou nekompletní, nekonzistentní, nedosažitelné nebo nejednoznačné, což jsou vlastnosti výše uváděné.

2. Standardy pro měření funkční velikosti softwaru

Existuje řada standardů, které popisují výpočet funkční velikosti softwaru. Níže je uveden soupis ISO norem, které se tomuto tématu věnují. Podrobně se všem těmto standardům však tato semestrální práce nevěnuje.

- FiSMA: ISO/IEC 29881:2010 Information technology – Systems and software engineering – FiSMA 1.1 functional size measurement method.
- **IFPUG: ISO/IEC 20926:2009 Software and systems engineering – Software measurement – IFPUG functional size measurement method.**

- Mark-II: ISO/IEC 20968:2002 Software engineering – Ml II Function Point Analysis – Counting Practices Manual
- Nesma: ISO/IEC 24570:2018 Software engineering – Nesma functional size measurement method version 2.3 – Definitions and counting guidelines for the application of Function Point Analysis
- **COSMIC: ISO/IEC 19761:2011 Software engineering. A functional size measurement method.**
- OMG: ISO/IEC 19515:2019 Information technology — Object Management Group Automated Function Points (AFP), 1.0

V následujících kapitolách této semestrální práce budou popsány normy ISO/IEC 20926:2009 (IFPUG) a ISO/IEC 19761:2011 (COSMIC FFP). Následně bude provedeno srovnání metod, které tyto normy popisují.

3. IFPUG

V 70. letech minulého století vzrostlo použití funkčních bodů jako měřítka funkční velikosti softwaru. Allan Albrecht byl první, kdo na tento nárůst zareagoval, vydal veřejně přístupnou metodu pro funkční dimenzování softwaru a nazval ji analýza funkčních bodů. V roce 1986 vznikla skupina International Function Point Users Group (IFPUG) a od svého vzniku neustále vylepšovala Albrechtovu metodu. Metoda měření funkční velikosti IFPUG je známá jako analýza funkčních bodů a její jednotky funkční velikosti se nazývají funkční body. (ISO, 2009)

Organizace mohou použít tento mezinárodní standard pro měření velikosti softwarového produktu k:

- podpoře analýzy kvality a produktivity
- odhadu ceny a zdrojů potřebných pro softwarový vývoj, údržbu a zlepšení
- zajištění normalizačního faktoru pro porovnání softwarů
- stanovení velikosti zakoupené aplikace pomocí funkčních velikostí všech funkcí, které jsou zahrnuty v balíčku
- pomoci uživatelům stanovit benefity aplikace pro jejich organizaci (ISO, 2009)

Analýza funkčních bodů měří software na základě kvantifikace úkolů a služeb (tj. funkčnosti) které software poskytuje uživateli. Cílem analýzy funkčních bodů je měřit:

- funkčnost implementovaná v softwaru, kterou uživatelé požadovali a dostali, využívají
- funkčnost ovlivněná vývojem, zlepšováním a údržbou softwaru nezávisle na technologii použité k implementaci (Struska, 2008)

Tato metoda se používá v počáteční fázi projektu. Pohlíží na aplikaci z úhlu uživatele. V počáteční fázi projektu určuje požadavky především byznys a je jasné, že lidé z této části nejsou znalí IT natolik, aby dokázali definovat své požadavky jinak, než jen sadou uživatelských požadavků a podmínek. Již na začátku procesu se proto abstrahuje od toho, jak bude aplikace implementována a jaké návrhové vzory budou použity.

3.1. Funkční bod

K určení velikosti funkčnosti se používají takzvané funkční body. Funkční bod si můžeme představit jako měrnou jednotku, na rozdíl však od fyzikálních měrných jednotek jako je kilogram nebo kilometr není tato jednotka tak exaktní. Je nutné, aby se jednotka práce dala přesně kvantifikovat. Toho dosáhneme díky využití jednotky, která měří požadovanou funkčnost, která je zadaná budoucím uživatelem. Požadované funkčnosti systému se dají složit v jednotlivé celky, které můžeme převést na funkční body. Mezi funkční body nezahrnujeme algoritmy, které jsou použité v kódu nebo například ve struktuře databáze. Tato metodika klade důraz na vstupní/výstupní soubory a transakce, vnitřní logické soubory jako jsou tabulky, kontrolní soubory nebo soubory s daty, toky informací, externí soubory a grafické rozhraní. (Struska, 2008)

3.2. Postup výpočtu

Tabulka 2: Postup výpočtu a výstupy z jednotlivých kroků (IFPUG, 2000)

KROK	**VÝSTUP**
Identifikace hranic systému	Jasně stanovené hranice
Identifikace funkcí systému	Zařazení jednotlivých funkcí do jedné z pěti komponent systému
Stanovení složitosti funkcí systému	Zařazení do kategorie
Stanovení hodnot pro datové funkce	Počet nekorigovaných funkčních bodů
Zhodnocení charakteristik vyvíjeného softwaru	Přidělení bodů jednotlivým charakteristikám systému
Stanovení hodnoty korekčního faktoru	Korekční faktor
Výpočet korigovaných funkčních bodů	Počet korigovaných funkčních bodů

3.2.1. Hranice systému

První krok k odhadu funkční velikosti je stanovení hranice systému. Pokud se jedná o systém, který nahrazuje systém předcházející, hranice se snadno určí podle hranic předchozího systému. Pokud se jedná o systém nový, ke stanovení hranic napomůže pohled na integraci aplikace v celkové byznys architektuře. Hranicemi se určí, co je uvnitř aplikace a co už je externí. (IFPUG, 2002)

3.2.2. Identifikace funkcí systému

Druhý krok je identifikace funkcí systému. Tato metoda dělí funkce na datové a transakční, které se následně rozdělují celkově na 5 hlavních komponent systému. Do datových funkcí patří vnitřní logické soubory (Internal Logical Files – ILF) a vnější soubory rozhraní (External Interface Files – EIF). Mezi transakční funkce poté patří vstupy do aplikace (External Inputs – EI), externí dotazy (External inQuiry – EQ) a výstupy z aplikace (External Outputs – EO). (IFPUG, 2002)

Tabulka 3: Rozdělení funkcí dle IPFUG (IPFUG,2000)

Datové funkce	Vnitřní logické soubory	ILF
	Vnější soubory rozhraní	EIF
Transakční funkce	Vstupy do aplikace	EI
	Externí dotazy	EQ
	Výstupy z aplikace	EO

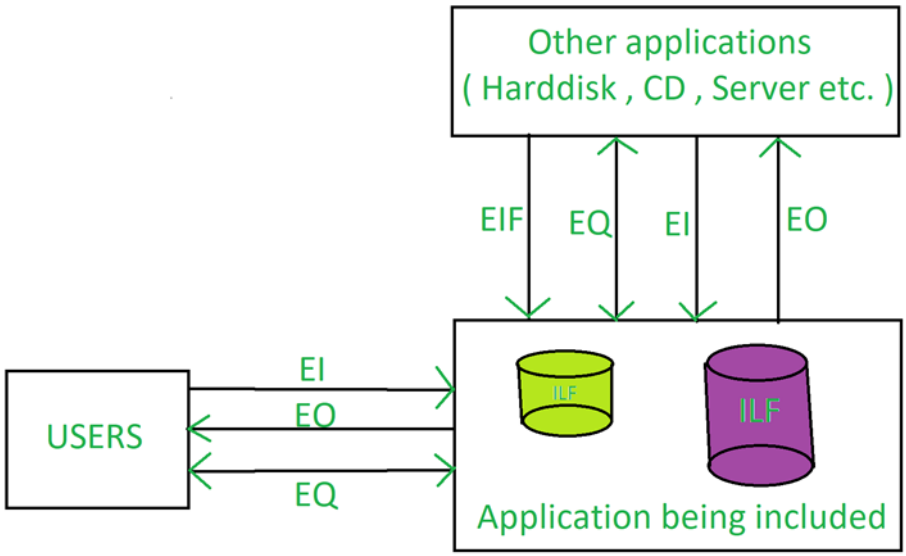
Vnitřní logické soubory jsou uživatelsky identifikované skupiny logicky souvisejících dat nebo řídících informací udržovaných v rámci hranic aplikace. Primárním záměrem vnitřních logických souborů je uchovávat data udržovaná prostřednictvím jednoho nebo více elementárních procesů. Příkladem takových souborů jsou například tabulky v relační databázi nebo například informace o ovládání aplikace – předvolby uživatelů, které aplikace ukládají. (IFPUG, 2000)

Vnější soubory rozhraní jsou uživatelsky identifikované skupiny logicky souvisejících dat nebo řídících informací, které jsou na rozdíl od vnitřních logických souborů uchovávána mimo hranice aplikace. Vnější soubory rozhraní jsou tedy vnitřní logické soubory spolupracující aplikace. Mezi takové můžeme řadit například vyskakovací nápovědy nebo soubory určené k importu. (IFPUG, 2000)

Vstupy do aplikace jsou základní procesy, které zpracovávají data nebo řídící informace, které přicházejí z vnějšku hranic aplikace. Primárním záměrem EI je udržovat jeden nebo více ILF a/nebo změnit chování systému. Mezi tyto řadíme například zadávání údajů uživateli, data nebo kanály souborů externími aplikacemi. (IFPUG, 2000)

Externí dotazy je základní proces, který odesílá data nebo řídící informace mimo hranice aplikace. Primárním záměrem externích dotazů je prezentovat informace uživateli prostřednictvím načítání dat nebo kontrolních informací z ILF, ELF. Logika zpracování neobsahuje žádné matematické vzorce nebo výpočty a nevytváří žádná odvozená data. Mezi externí dotazy můžeme řadit například sestavy vytvořené počítanou aplikací, kde sestava neobsahuje žádná odvozená data, výstup z databáze údajů o zákaznících seřazených dle jména. (IFPUG, 2000)

Výstupy z aplikace je základní proces, který odesílá data nebo řídící informace mimo hranice aplikace. Primárním záměrem výstupů z aplikace je prezentace informace uživateli prostřednictvím procesní logiky, tj. pomocí matematických vzorců a výpočtů. Mezi výstupy z aplikace řadíme například sestavy vytvořené počítanou aplikací, kde sestavy obsahují odvozené informace, tj. například upozornění na stav systému. (IFPUG, 2000)



Obrázek 2: Softwarová aplikace z pohledu Function Point practitioner (GeeksforGeeks, 2019)

3.2.3. Stanovení složitosti funkcí systému

Ve třetím kroku se vychází z identifikovaného rozdělení z kroku číslo 2. Složitost se určí pomocí systémových charakteristik, které jsou specifické pro každou skupinu. Mezi tyto charakteristiky patří (IFPUG, 2000):

Referenční soubory (File Type Referenced – FTR) – vnitřní logický soubor, který je čten nebo udržován transakční funkcí nebo vnější soubor rozhraní, který je čten transakční funkcí

Datové prvky (Data Element Type – DET) – jedinečné, uživatelsky rozpoznatelné a neopakující se pole

Záznamové prvky (Record Element Type – RET) – skupina uživatelsky rozpoznatelných datových prvků v rámci vnitřních logických souborů nebo vnějších souborů rozhraní

Tabulka 4: Rozdělení komponent dle IFPUG (IFPUG, 2000)

Komponenty	**Systémové charakteristiky**		
	FTR	**DET**	**RET**
Vnitřní logické soubory		X	X
Vnější soubory rozhraní		X	X
Vstupy do aplikace	X	X	
Externí dotazy	X	X	
Výstupy z aplikace	X	X	

Poté se spočítá pro jednotlivé komponenty systému počet příslušných systémových charakteristik (např. pro vnitřní logické soubory počet datových prvků a záznamových prvků) a podle škály, která je uvedená v tabulkách níže se ohodnotí příslušným stupněm jako *nízký, průměrný* nebo *vysoký*.

Tabulka 5: Hodnocení pro vnitřní logické soubory a externí rozhraní (IPFUG, 2000)

Referenční soubory	**Datové prvky**		
	1–19	**20–50**	**> 50**
0–1	Nízký	Nízký	Průměrný
2–5	Nízký	Průměrný	Vysoký
> 5	Průměrný	Vysoký	Vysoký

Tabulka 6: Hodnocení pro vstupy do aplikace (IPFUG, 2000)

Referenční soubory	**Datové prvky**		
	1–4	**5–15**	**> 15**
0–1	Nízký	Nízký	Průměrný
2	Nízký	Průměrný	Vysoký
> 2	Průměrný	Vysoký	Vysoký

Tabulka 7: Hodnocení pro výstupy z aplikace a uživatelské dotazy (IPFUG, 2000)

Referenční soubory	**Datové prvky**		
	1–5	**6–19**	**> 19**
0–1	Nízký	Nízký	Průměrný
2–3	Nízký	Průměrný	Vysoký
> 3	Průměrný	Vysoký	Vysoký

3.2.4. Stanovení hodnot pro datové funkce

Ve čtvrtém kroku se z hodnot *nízký, průměrný* a *vysoký* stanou číselné hodnoty pomocí přepočítávání na počet nekorigovaných funkčních bodů. Na základě tabulek níže se počty jednotlivých komponent znásobí stanovenou konstantou.

Tabulka 8: Konstanty pro transakční procesy (IPFUG, 2000)

Odhad	Hodnoty		
	Vstupy do aplikace	Externí dotazy	Výstupy z aplikace
Nízký	3	3	4
Průměrný	4	4	5
Vysoký	6	6	7

Tabulka 9: Konstanty pro datové funkce (IPFUG, 2000)

Odhady	Hodnoty	
	Vnitřní logické soubory	Vnější logické soubory
Nízký	7	5
Průměrný	10	7
Vysoký	15	10

Následně se hodnoty dílčích výsledků sečtou a dostaneme celkový počet nekorigovaných funkčních bodů celé softwarové aplikace, které značíme UFP (Unadjusted Function Point count). (IFPUG, 2000)

3.2.5. Zhodnocení charakteristik vyvíjeného softwaru

V pátém kroku se ohodnocují nefunkční požadavky systému. V tabulce níže je uvedeno 14 charakteristik. Tyto charakteristiky označují složitost vyvíjeného systému bez ohledu na podmínky, ve kterých bude systém vyvíjen. Ty se ohodnotí na stupnici od 0 do 5 podle míry uplatnění příslušného faktoru (0 – žádný vliv, 5 – velmi významný vliv). Toto ohodnocení se značí jako DI (Degrees of influence). Suma všech DI se označuje jako TDI (Total Degree of influence). (IFPUG, 2000)

Tabulka 10: Hodnocení nefunkčních požadavků systému (IPFUG, 2000)

Obecné charakteristiky systému		**Krátký popis**	**Bodové ohodnocení**
1	Datová komunikace	Kolik komunikačních zařízení podporuje přenos nebo výměnu informací s aplikací nebo systémem?	
2	Distribuované zpracování dat	Jak jsou zřízena distribuovaná data a jejich zpracování?	
3	Výkon	Byla časová odezva nebo výkon v souladu s požadavky uživatele?	
4	Intenzita využití konfigurace	Jaká je intenzita využití současné hardwarové platformy, na které budou aplikace provozovány?	
5	Transakční míra	Jak často jsou transakce zpracovávány – denně, týdně, měsíčně atd.?	
6	On-line vkládání dat	Jaké procento informací je vkládáno on-line?	

7	Výkonnost koncového uživatele	Byla aplikace navržena, aby zlepšila pracovní výkon koncového uživatele?	
8	On-line aktualizace	Kolik vnitřních logických souborů je aktualizováno on-line?	
9	Složité zpracování	Disponuje aplikace rozsáhlým logickým a matematickým zpracováním?	
10	Znovupoužitelnost	Byla aplikace vyvinuta, aby uspokojila jednu nebo více uživatelských potřeb?	
11	Jednoduchost instalace	Jak složitá je úprava a instalace?	
12	Provozní jednoduchost	Jak účinně a/nebo automatizovaně je aplikace startována, zálohována a obnovena?	
13	Multifunkční využití	Byla aplikace speciálně navržena, vyvinuta a podporována, aby mohla být instalována pro multifunkční využití v rámci organizací?	
14	Ulehčení změn	Byla aplikace speciálně navržena a vyvinuta, aby podporovala lehké zavedení změn?	
Součet bodů charakteristik systému (TDI)			

3.2.6. Stanovení hodnoty korekčního faktoru

V šestém kroku se vypočte hodnota korekčního faktoru. Využijeme k tomu hodnotu TDI, kterou jsme získali v předešlém kroku. Hodnotu TDI vynásobíme číslem 0,01 a následně k ní přičteme konstantu 0,65. Tímto získáme korekční faktor VAF (Value Adjustment Factor). (IFPUG, 2000)

$$VAF = TDI \cdot 0{,}01 + 0{,}65$$

3.2.7. Výpočet korigovaných funkčních bodů

V posledním, sedmém kroku se vypočte celkový počet korigovaných funkčních bodů. Tuto hodnotu získáme vynásobením nekorigovaných funkčních bodů (UFP) a korekčního faktoru (VAF) a hodnotu označíme AFP (Adjusted Function Point). (IFPUG, 2000)

$$AFP = UFP \cdot VAF$$

4. COSMIC

Common Software Measurement International Consortium tedy zkráceně COSMIC je sdružení, jehož vznik se datuje k roku 1998, kdy došlo k neformálnímu setkání členů WG12[1] v Londýně, kde se rozhodli vyvinout FSM[2] metodu druhé generace. Rozdíl oproti první generaci FSM měl spočívat v tom, že druhá generace by pouze vycházela ze základních zavedených principů softwarového inženýrství na rozdíl od té první, která používala ad-hoc model softwarové funkcionality. Skupina mohla čerpat z minulých zkušeností s metodami FSM a od počátku se snažila vyhovět normě ISO/IEC 14143/1:1997[3]. Metoda by měla být stejně použitelná pro celou paletu různých druhů software od byznys aplikací až po infrastrukturní aplikace. (COSMIC, 2020)

První verze ‚COSMIC-FFP v2.0' vznikla v říjnu 1999. V letech 2000 a 2001 proběhly první úspěšné rozsáhlé pokusy v praxi. COSMIC zveřejnil verzi 2.2 metody v lednu 2003 a verzi 3.0 v září 2007, kdy

[1] WG (Working Group), je pracovní skupina Mezinárodní organizace pro normalizaci (ISO)
[2] FSM (Functional Size Measurement) je proces měření funkční velikosti.
[3] ISO (International Organization for Standardization) je v překladu Mezinárodní organizace pro normalizaci.

s touto verzí přichází nový název ‚COSMIC method'. Následující verze mají několik dalších vylepšení, avšak od jejich prvního zveřejnění nedošlo k žádným změnám základních principů metody. (COSMIC, 2020)

V důsledku rozhodnutí, výboru ISO/IEC JTC1/SC7, nechat rozhodnout trh byly v průběhu roku 2002/3 zveřejněny mezinárodní normy pro metodu COSMIC (ISO/IEC 19761), metodu IFPUG – functional size component (ISO/IEC 20926), metodu MkII FPA (ISO/IEC 20968) a metody NESMA (ISO/IEC 24570). Norma ISO/IEC 19761 se každých několik let reviduje, aby byla sladěna s nejnovější verzí metody COSMIC. (COSMIC, 2020)

Dle výroční zprávy (COSMIC, 2015) z roku 2014 byla verze COSMIC povýšena na verzi 4.0 a větší přístup mají i nováčci a ne-anglicky hovořící uživatelé. Dále byla vytvořena on-line certifikační zkouška pro *entry-level* (vstupní) certifikaci, díky čemuž mohou uživatelé COSMIC prokazovat jejich základní dovednosti.

4.1. ISO 19761:2011

Standard ISO 19761:2011 je aktualizací verze z roku 2003, která byla založena na publikaci COSMIC-FP 2.1 vydané Mezinárodním konsorciem pro měření softwaru (Common Software Measurement International Consortium). Tato publikace se zaměřovala na použitelnost pro business, real-time a systémový software. Norma 19761:2011 si klade za cíl řešit potřeby dodavatelů softwaru, jejichž požadavkem je převádění požadavků zákazníků na funkční velikost softwaru, od které se dále odvíjí výpočet nákladů. Druhou skupinou uživatelů této normy jsou zákazníci, kteří si díky měření velikosti softwaru mohou ověřovat výkon dodavatele. Hlavním důvodem pro vznik této normy je potřeba měřit funkční velikost softwaru v souvislosti s plánováním nákladů v podniku. (ISO, 2011)

Tato norma obsahuje definice, konvence a metody pro měření funkční velikosti aplikačního a real-time softwaru nebo jejich kombinace na principu COSMIC. Použití této normy se nehodí v případě softwaru obsahujícího složité matematické algoritmy, simulace, proces učení a zpracování zvuků čí videozáznamů. (ISO, 2011)

4.2. Metoda COSMIC

Metoda COSMIC se využívá k měření funkční velikosti softwaru a je dostupná veřejně a zdarma. Lze pomocí ní měřit jednotlivé požadavky uživatele na funkcionalitu softwaru či celé systémy. Tato metoda je postavené na základě funkčních bodů COSMIC, které slouží jako měřítko množství uživatelských funkcí softwaru. Rozlišujeme dva typy požadavků zákazníka – požadavky FUR (functional user requirements = funkční požadavky uživatele), tedy co software má dělat a požadavky, které nejsou FUR, tedy požadavky, jak to má software dělat, kam řadíme například vzhled uživatelského prostředí nebo rychlost načítání dat. Protože není vždy lehké tyto dva druhy požadavků rozdělit, lze v COSMIC postupovat dle jednotlivých softwarových artefaktů (pro příklad rozvržení obrazovky, případové diagramy, návrh databáze, diagram entit). Z těchto artefaktů je možné extrahovat další možnou funkcionalitu nad rámec požadavků zákazníka. Důležitou vlastností metody COSMIC je, že není vhodná pro měření nefunkčních požadavků. (Symons, 2020)

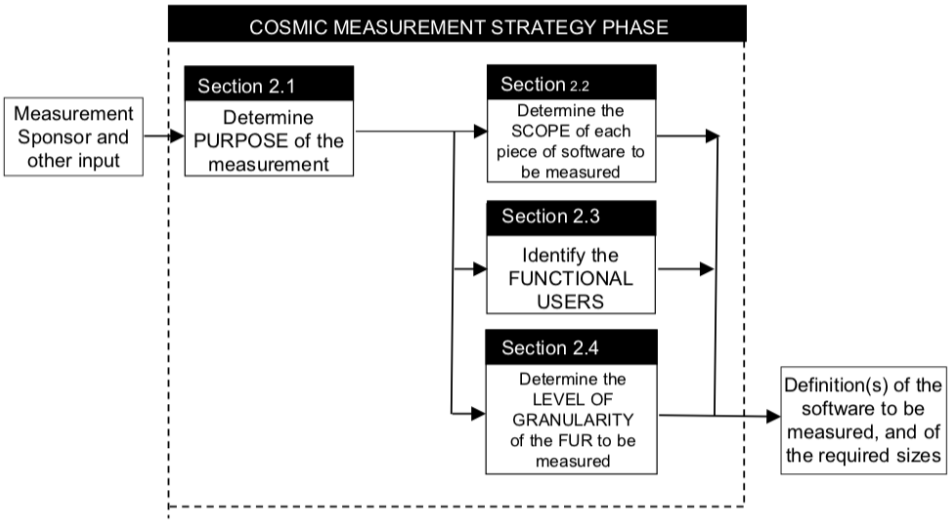
4.3. Proces měření

Proces měření softwaru dle COSMIC se skládá ze tří fází: strategie měření, mapování a měření. Tyto fáze jsou blíže popsány v následujících podkapitolách.

4.3.1. Strategie měření

Do fáze strategie měření vstupují požadavky zákazníka. Jako první krok je potřeba definovat účel měření a použití jeho výsledků. Toto se odvíjí od požadavků sponzora. Dále je definován rozsah (scope)

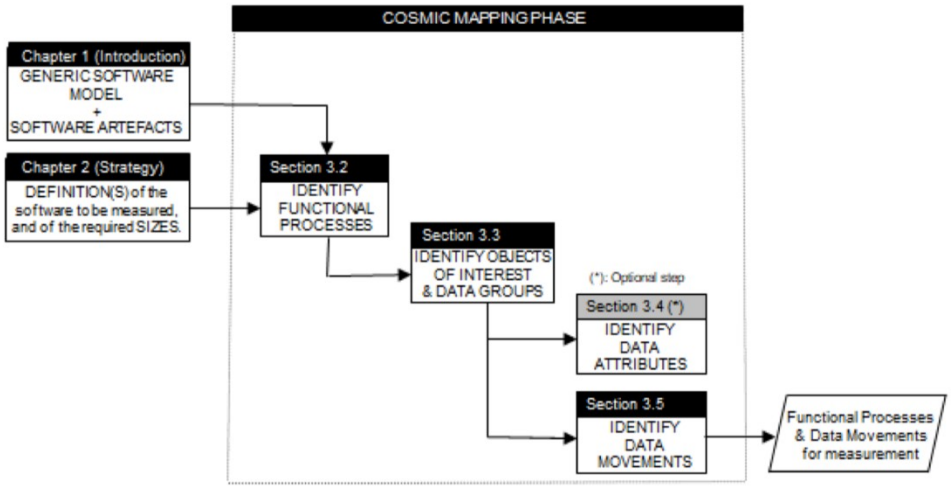
měření, který pokrývá jednotlivé vrstvy architektury softwaru tak, že pro každou vrstvu se určí samostatný rozsah měření. Je nutné identifikovat uživatele funkcionalit (functional users), určit úroveň granularity funkčních požadavků zákazníka (FUR) a vybrat vzory pro strategii měření, pokud již v organizaci existují. Jako výstup této fáze je definováno, co budeme měřit. (Symons, 2020)



Obrázek 3: Fáze strategie měření (Symons, 2020)

4.3.2. Mapování

Do fáze mapování vstupuje generický model softwaru a softwarové artefakty a dále pak výstup s předchozí fáze strategie měření. Tato fáze slouží k identifikaci funkčních procesů a pohybu dat z a do artefaktů softwaru. Patří sem mapování artefaktů na generický softwarový model, dále identifikace funkčních procesů, datových atributů, principů pohybu dat a identifikace konceptů COSMIC v již existujících artefaktech softwaru. V této fázi se definují spouštěcí události (triggering events) a spouštěné funkční procesy softwaru, které mohou být zahájeny uživatelem nebo systémem. Pohyb dat je popsán pomocí pohybu dat (data movements) a manipulace s daty (data manipulation), které pracující se skupinami dat (data groups). (Symons, 2020)



Obrázek 4: Fáze mapování (Symons, 2020)

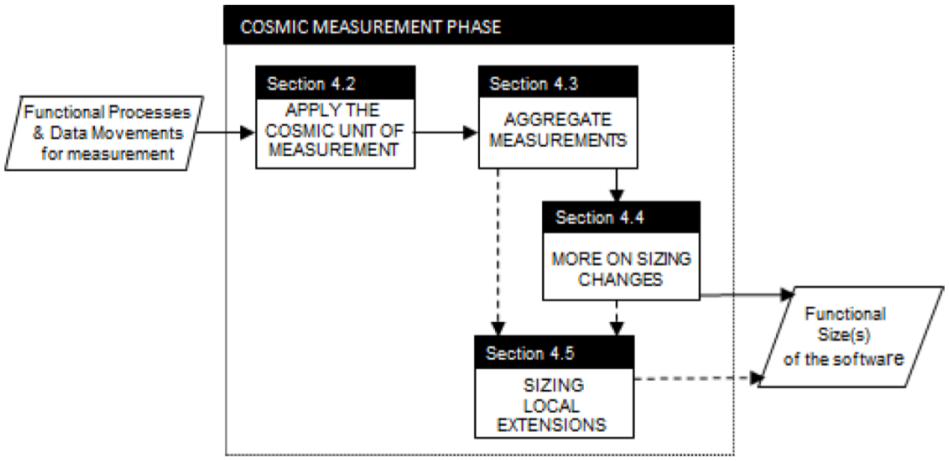
Typy pohybu dat jsou:

- vstup (entry) data od uživatele k softwaru
- výstup (exit) data od softwaru ven
- psát (write) data do trvalého úložiště
- číst (read) data z trvalého úložiště

Data groups mohou mít definované atributy tak, aby byly jedinečné. Datová skupina vždy pokrývá jeden uživatelská zájem (object of interest). Výstupem fáze mapování jsou identifikovaná funkční procesy a pohyby dat, které budeme v poslední fázi měřit. (Symons, 2020)

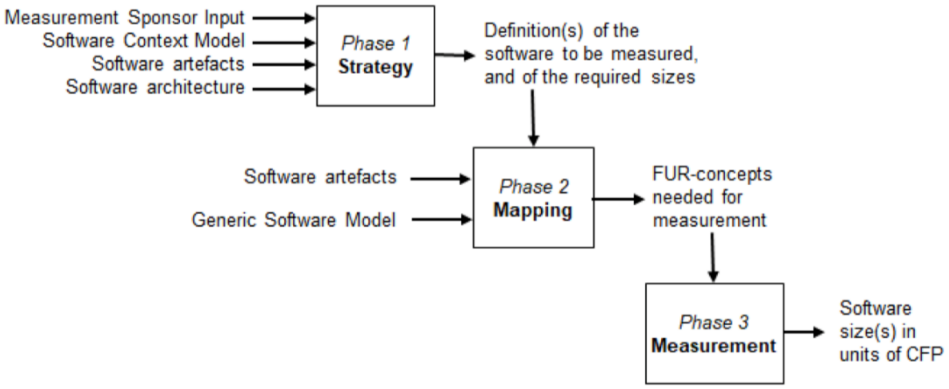
4.3.3. Měření

Měření je poslední fází procesu a dochází zde k výpočtu velikosti softwaru za použití COSMIC funkčních bodů (CFP = COSMIC Functional Points). Jeden funkční bod se rovná jednomu pohybu dat. Velikost funkčních procesů se pak spočítá jako: suma Entries (vstupů) + suma Exits (výstupů) + suma Reads (čtení) + suma Writes (zapisování). Velikost změn ve funkčních procesech se pak vypočítá pomocí: suma velikosti nových pohybů dat + suma velikosti modifikovaných pohybů dat + suma velikosti smazaných pohybů dat. Výstupem této finální fáze je pak funkční velikost softwaru vyjádřená v COSMIC funkčních bodech. (Symons, 2020)



Obrázek 5: Fáze měření (Symons, 2020)

Celý proces měření můžeme vidět na obrázku níže. Jednotlivé fáze jsou podrobněji rozebrány v předchozích podkapitolách.



Obrázek 6: COSMIC proces měření (Symons, 2020)

4.4. Možná rozšíření

Na metodu COSMIC je vzhledem k jejímu volnému šíření možno aplikovat rozšíření dle potřeby konkrétní organizace. Lze například využít lokální rozšíření pro měření některých aspektů funkčních požadavků ve větším detailu. COSMIC může být přínosný i u měření systémů s manipulaci s daty. Výsledky této metody lze také kombinovat s dalšími měřeními, protože její výsledky nezohledňují veškeré faktory ovlivňující složitost softwaru. Existuje také rozšíření pro práci se softwarem obsahujícím složité algoritmy. (Symons, 2020)

5. Srovnání metod IFPUG FPA a COSMIC FFP

Metody měření pomocí funkčních bodů IFPUG FPA a COSMIC-FFP mají mnoho podobných vlastností. Mezi ně patří:

- Uznání elementárních procesů jako funkčních jednotek pro měření
- Základní jednotkou pro definování funkčních uživatelských požadavků je Base Functional Component (základní komponent funkcionality)
- Vnímají přesuny dat, jakou součást funkční velikosti softwaru
- Do funkční velikosti se počítají datové přístupy k trvalým datům
- Obě specificky neměří velikost algoritmů, procesní logiky, transformace dat a složitých výpočtů
- Dokáží přibližně odhadnout velikost softwaru již v raných fázích vývoje

(Santillo, 2012)

V následující tabulce jsou zaznamenány hlavní rozdíly mezi metodami IFPUG FPA a COSMIC-FFP. Tyto rozdíly jsou rozděleny barevně do třech kategorií. Zelenou jsou označené technické rozdíly metod, červenou rozdíly ve vnímání dat a jejich pohybu, modrá pak značí obecné rozdíly v dostupnosti a použití metod.

IFPUG FPA	**COSMIC-FFP**
Pouze procesy	Subprocesy
Sdílení dat napříč procesy	Data samostatně v každém procesu
Měří procesy a data zvlášť	Vše dohromady
Práce s vícevrstvou architekturou není explicitně definována	Pravidla pro vícevrstvou architekturu
Specifikace funkcionality nemusí být detailní, můžeme volit rozsah komplexity	Potřeba detailních požadavků na funkcionalitu
Měření z pohledu koncového uživatele	Měření z pohledu koncového uživatele a vývojáře
Bere v potaz technické požadavky a požadavky na kvalita (prostřednictvím Value adjustment factor)	Měří pouze na základě funkčních požadavků uživatele
Technické požadavky, požadavky na kvalitu a funkční požadavky	Funkční požadavky
Data Element Type (DET), Record Element Type (RET), File Type Referenced (FTR)	Data group (datová skupina), Datový atribut
External imput, External output, External inquiry, Internal logical file, external interface file	Entry, Exit, read, write
ISO 20296: 2009	ISO 19761:2011
Individuální členství - 185 dolarů/rok	Dostupné zdarma

Cca 390 stran	Cca 75 stran
Certifikace: Certified Function Point Fellow Certified Function Point Specialist Certified Function Point Prectitioner Certified SNAP Practicionair	Certifikace pro softwarové inženýry
Závislé na technologie vývoje softwaru a programovacím jazyce	Nezávislé na technologie vývoje softwaru a programovacím jazyce
Vhodné pro business aplikační software	Vhodné pro Real-time software

Obrázek 7: Výsledné porovnání metod, zdroj autor na základě (Total metrics, 2003)

I když se jedná o dvě metody, který mají stejný cíl, tedy měření funkční velikosti softwaru, v mnohém se liší. IFPUG FPA metoda byla vydána před publikací mezinárodní standardu ISO/IEC 14143, což činí její kompatibilitu s touto normou náročnější. Na druhé straně metoda COSMIC-FFP byla vydána už se znalostí této ISO normy a brala budoucí kompatibilitu v potaz. Rozsah softwarových specifikací, které lze měřit pomocí COSMIC-FFP je značně širší než u metody IFPUG FPA. Hlavní rozdílem v jejich dostupnosti je fakt, že podklady pro COSMIC metodu jsou zdarma a volně přístupné. Na obě metody existuje certifikace, přičemž IFPUG nabízí několik různých certifikačních stupňů. (Santillo, 2012)

Závěr

V 70. letech minulého století vzrostlo použití funkčních bodů jako měřítka funkční velikosti softwaru. Allan Albrecht byl první, kdo na tento nárůst zareagoval, vydal veřejně přístupnou metodu pro funkční dimenzování softwaru a nazval ji analýza funkčních bodů. Od té doby vzniklo několik metod, které na Albrechtovu práci navázali. Tato semestrální práce se zabývala metodami odhadu funkční velikosti softwaru právě na základě funkčních bodů a jejich porovnáním.

Nejprve byl v práci čtenáři ze široka popsán pojem požadavek zákazníka na software, který je vstupem pro měření funkční velikosti. V dalších kapitole byla představena norma ISO 20926:2009, jejímž autorem je společnost IFPUG a popsán výpočet funkční velikosti softwaru, který je součásti uvedené normy. V následující kapitole byla popsána norma ISO 19761:2011 a k ní příslušná metoda COSMIC-FFP od společnosti COSMIC. V poslední kapitole této semestrální práce byly tyto dvě normy porovnány.

Hlavním cílem této práce bylo srovnání metod odhadu funkční velikosti pomocí funkčních bodů a toho bylo dosaženo v kapitole Srovnání metod IFPUG FPA a COSMIC FFP. Dílčích cílů, které předcházejí cíli hlavnímu bylo dosaženo v samostatných kapitolách. Jedná se o představení metod měření velikosti softwaru IFPUG v kapitole a COSMIC v kapitole. Dalšího dílčího cíle, a to představení ISO norem pokrývající tyto metody bylo dosaženo v rámci kapitol o jednotlivých metodách. Pro dosažení všech cílů jsme analyzovali dostupné elektronické zdroje.

Metoda COSMIC-FFP je na rozdíl od metody IFPUG FPA dostupná zdarma a je výrazně méně obsáhlá. Z toho usuzujeme, že by mohla být vhodnější pro menší a začínající podniky. Výhodou metody COSMIC–FFP je, že bere v potaz navíc pohled vývojáře a je nezávislá na technologii vývoje. Vzhledem k tomu, že metoda IFPUG FPA je vhodná pro business aplikační software a nabízí několik stupňů certifikací lze usuzovat, že bude ideálním řešením pro korporátní prostředí. Možným rozšířením této práce je rozbor dalšího užití těchto metod a jejich modifikace tak, aby je bylo možno použít i při měření dalších typů softwaru, které obsahují například složité matematické algoritmy apod.

Literatura

AMBLER, W. Scott, 2020. Examining the Agile Cost of Change Curve. *Agile Modeling* [online]. Ambysoft Inc. [vid. 2020-12-17]. Dostupné z: http://www.agilemodeling.com/essays/costOfChange.htm

COSMIC, 2015. Annual Report 2014. *SlideShare.net* [online]. [vid. 2020-12-17]. Dostupné z: https://www.slideshare.net/COSMIC-FSM/cosmic-annual-report-2014

COSMIC, 2020. History. *COSMIC The standart methodology for sizing software* [online]. [vid. 2020-12-17]. Dostupné z: https://cosmic-sizing.org/cosmic-sizing/functional-size-measurement/history/

COSMIC-FFP and IFPUG 4.1: Similarities and Differences. Totalmetrics.com [online]. [vid. 2020-11-14]. Dostupné z: https://www.totalmetrics.com/function-point-resources/downloads/COSMIC-Versus-IFPUG-Similarities-and-Differences.pdf

DAVIS, M. Alan, 2005. Just Enough Requirements Management: Where Software Development Meets Marketing. New York: Dorset House Publishing. 240 s. ISBN 978-0932633644.

Examining the Agile Cost of Change Curve. Agilemodeling.com [online]. [vid. 2020-12-10]. Dostupné z: http://www.agilemodeling.com/essays/costOfChange.htm

Function Point Counting Practices Manual, release 4.1.1, *International Function Point Users 2000*, USA, ISBN 0-963-1742-7-4

GeeksforGeeks, *Software Engineering | Functional Point (FP) Analysis*. [online]. 2019 [vid. 2020-11-15]. Dostupné z: https://www.geeksforgeeks.org/software-engineering-functional-point-fp-analysis/

IEEE, 1990. IEEE Standard Glossary of Software Engineering Terminology. New Jersey: IEEE Std. ISBN 9781559370677

International Function Points Users Group. *IT Measurement PractialAdvice from the Experts*. Addison-Wesley. Boston 2002. ISBN 0-201-74158-X.

ISO/IEC 19761:2011. Iso.org [online]. [vid. 2020-11-14]. Dostupné z: https://www.iso.org/obp/ui/#iso:std:iso-iec:19761:ed-2:v1:en

ISO. *International Organization for Standardization.* [online]. 2009 [vid. 2020-11-15]. Dostupné z: http://www.iso.org/iso/catalogue_detail.htm?csnumber=51717

LAWRENCE, Brian. 1997. Requirements Happens… Cutter Business Technology Journal. 10(4), 3-9. ISSN 2475-3718.

SANTILLO, Luca, 2012. Easy Function Points -- 'Smart' Approximation Technique for the IFPUG and COSMIC Methods. In: 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement [online]. ISBN 978-1-4673-3127-2. Dostupné z: doi:10.1109/IWSM-MENSURA.2012.29

SAWYER, Pete a SOMMERVILLE, Ian, 1997. Requirements Engineering: A Good Practice Guide. New York: Wiley. 404 s. ISBN 978-0471974444.

STRUSKA, Zdeněk. *Metody odhadu složitosti vývoje moderního softwaru.* Borovany, 2008. Disertační práce. Česká zemědělská univerzita v Praze – Provozně ekonomická fakulta, Katedra informačního inženýrství. Vedoucí disertační práce Jiří Vaníček.

SYMONS, Charles, 2020. A Guide to Software Size A Guide to Software Size Measurement [online]. [vid. 2020-11-14]. Dostupné z: https://cosmic-sizing.org/wp-content/uploads/2020/08/Measuring-software-size-v1.0-August-2020.pdf

WIEGERS, E. Karl a BEATTY, Joy, 2013. Software Requirements. 3. vydání. Redmont: Microsoft Press. 637 s. ISBN 9780735679665.