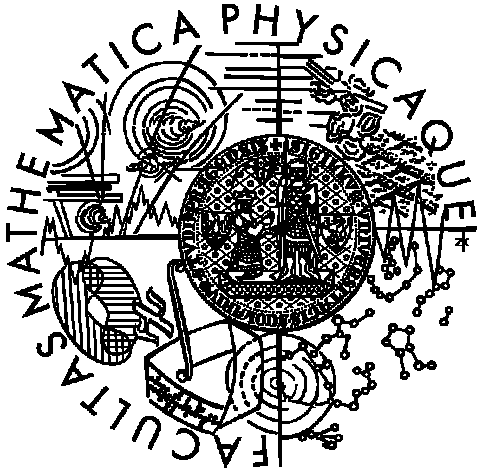
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

# DIPLOMOVÁ PRÁCE




*Bc. Jan Cipra*

**Metody pro odhadování pracnosti softwarových projektů ve vývoji**

*Katedra softwarového inženýrství*
Vedoucí diplomové práce: *doc. Ing. Karel Richta, CSc.*
Studijní program: *Informatika, Softwarové inženýrství*

Poděkování:

Na tomto místě bych rád poděkoval vedoucímu mé diplomové práce doc. Ing. Karlu Richtovi, CSc. za odborné vedení, cenné rady a připomínky, které mi poskytl. Dále bych chtěl poděkovat Ing. Jiřímu Koutníkovi vedoucímu J2EE Systems Komerční Banky za možnost práci zpracovat ve spolupráci s KB a za cenné připomínky.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne *Jan Cipra*

**Název práce**: Metody pro odhadování pracnosti softwarových projektů ve vývoji

**Autor**: Bc. Jan Cipra

**Katedra**: Katedra softwarového inženýrství

**Vedoucí diplomové práce**: doc. Ing. Karel Richta, CSc.

**E-mail vedoucího**: richta@fel.cvut.cz

**Abstrakt**:

*Práce se zabývá odhadováním softwarových projektů. V první části jsou představeny vybrané metody pro odhadování a problémy s odhadováním spojené. Následuje analýza stavu odhadování a volba metod vhodných pro použití v Komerční bance.*

*V druhé části práce je navrhnuta metodika pro odhadování velikosti softwarových projektů a práce potřebné na jejich dokončení. Tato metodika je určena nejen pro vývoj v Komerční bance, s jejíž spoluprací práce vznikla, ale je s menšími úpravami dostatečně universální a použitelná i v jiných společnostech. Součástí práce je také jednoduchý nástroj určený pro podporu odhadování pomocí metod z navržené metodiky.*

**Klíčová slova**: *projektové řízení, odhadování pracnosti, softwarové projekty, metodika*

**Title**: Methodology for Effort Estimation of Software Development Process

**Author**: Bc. Jan Cipra

**Department**: Department of Software Engineering

**Supervisor**: doc. Ing. Karel Richta, CSc

**Supervisor's e-mail address**: richta@fel.cvut.cz

**Abstract**:

*The work deals with software projects estimation. The first part presents selected methods for estimating and associated problems. Next there is an analysis of the estimation and the choice of methods suitable for use in the Komerční banka.*

*In the second part of the work, there is proposed a methodology for estimating the size of software projects and the work needed for their completion. This methodology is not intended only for the software development in the Komerční banka, with whose cooperation the work was created, but it is with minor changes sufficiently universal and applicable in other companies. Part of this work is also a simple tool designed to support the estimation using the methods of the proposed methodology.*

**Keywords**: *project managing, effort estimation, software projects, methodology*

# OBSAH

# 1 Úvod

Přesné odhadnutí projektu z hlediska pracnosti a délky trvání je důležitým předpokladem pro jeho správné naplánování. Na základě odhadu se rozhodujeme o tom, zda projekt budeme realizovat, případně v jakém rozsahu. Odhad projektu v jeho prvotní fázi slouží pro prioritizaci různých projektů v rámci firmy vzhledem k dostupným kapacitám. Určujeme na jeho základě také datum dokončení projektu, na které mohou být následně navázány například určité marketingové cíle či smluvní závazky. Proto každé zpoždění projektu (oproti předpokládanému plánu) způsobené nepřesným odhadem nebo neuvědoměním si jeho limitované přesnosti a z toho vyplývající nedostatečné rezervy v plánu, generuje nemalé náklady nejen z prodlužujícího se projektu, ale také z neplnění závazků. Z čehož vyplývá i ztráta prestiže a potenciálních zákazníků. Je nutné si tedy uvědomit, že správné odhadnutí projektu mnohdy přímo implikuje jeho včasné dokončení a naopak. Podhodnocení pracnosti projektu během plánování například o 40% neznamená, že celkové náklady po dokončení se budou lišit rovněž o 40%. Obvykle se konečné náklady liší o násobek původní odchylky v pracnosti.

I přes všechny tyto hrozby je odhadování softwarových projektů v mnoha firmách nedoceněnou součástí projektového řízení a není mu věnována dostatečná pozornost. Odhady jsou často tvořeny bez jakékoliv metodiky, pro každý projekt jiným způsobem a vytváří je většinou architekti či přímo vývojáři. V těchto případech se navíc ani nesleduje přesnost odhadů a nesbírají se data z projektů, na základě kterých bychom mohli následně metody odhadování zpřesňovat.

Pro minimalizaci rizika překročení rozpočtu je důležité mít proces odhadování pod kontrolou, být si vědom přesnosti odhadů, které jsme schopni v jednotlivých fázích projektu dosáhnout, a vzít tuto přesnost v potaz při dalším plánování

Zlepšení metod závisí na poctivém shromažďování dat o projektech, na přesnosti odhadů a na následném použití těchto dat pro úpravu stávajících metod.

Následující text nastíní čtenáři přehled o odhadovacích metodách pro jednotlivé fáze rozpracovanosti projektu a přiblíží, jak je možné nastavit komplexní metodiku pro odhadování softwarových projektů.

## 1.1 Cíl práce

Hlavním cílem této práce je navrhnout metodiku pro odhadování velikosti softwarových projektů a práce potřebné na jejich dokončení. Tato metodika by měla být určena nejen pro vývoj v Komerční bance (dále jen KB), s jejíž spoluprací tato práce vznikla, ale měla by být s menšími úpravami dostatečně universální a použitelná i v jiných společnostech.

Dalším cílem práce je navrhnout a implementovat jednoduchý nástroj, který bude obsahovat databázi dokončených a běžících projektů, rozhraní pro sběr těchto dat a také podporu pro odhadování pomocí metod z navržené metodiky.

Posledním úkolem je provedení případové studie na dokončených projektech v Komerční bance.

## 1.2 Struktura práce

Za účelem naplnění výše zmíněných cílů se v první části práce zaměřím na úvod do problematiky odhadování projektů a na některé poznatky, které s tím úzce souvisí. Nastíním též koncept fungování odhadování a problémy, které je potřeba řešit.

Z nepřeberné škály metod vhodných pro odhadování velikosti a pracnosti softwarových projektů uvedu stručný přehled některých z nich. U každé metody se budu snažit stručně popsat její vlastnosti s tím, že se zaměřím na hlavní klady, zápory a použitelnost s ohledem na velikost projektu a fázi, ve které se projekt nachází.

Dalším bodem bude provedení analýzy současného stavu odhadování projektů v Komerční bance. Pokusím se zmapovat požadavky na metodiku, která by měla odhadování nejen zpřesnit a standardizovat, ale také se snažit o maximální jednoduchost a intuitivnost. Provedu SWOT analýzu stávajících metod pro odhadování. Na základě zjištěných potřeb porovnám metody uvedené v první části práce s ohledem na vhodnost použití v KB.

Hlavní částí práce a současně splněním hlavního cíle práce je návrh metodiky pro odhadování softwarových projektů. Tuto metodiku vypracuji v kontextu projektové metodiky KB, což by ovšem nemělo bránit použití metodiky i v jiných společnostech (za předpokladu několika menších úprav).

Na metodiku pak navazuje specifikace databáze pro sběr dat o projektech a nástroje pro podporu odhadování. Uvedu i popis uživatelského prostředí tohoto nástroje a postupy práce při odhadování a zadávání dat.

V poslední části práce provedu případovou studii navrhované metodiky na několika již dokončených projektech v KB.

# 2 Úvod do problematiky

V této kapitole popíši několik bodů, které jsou důležité pro odhadování softwarových projektů. Nejprve to budou problémy, na které můžeme během odhadování narazit, a posléze se zaměřím na metody určování velikosti projektu.

## 2.1 Problémové body

Nejdříve proberu problémy spojené s odhadováním a jejich možná řešení. Prvním problémem je vztah pracnosti a velikosti projektu a dalším pak problém podhodnocování či nadhodnocování odhadů.

### 2.1.1 Nelinearita závislosti nárůstu pracnosti na velikosti projektu

Takzvaná „nevýhoda velikosti“ projektu spočívá v tom, že na vývoj projektu, který je 10 krát větší než jiný projekt, bude typicky potřeba více jak desetinásobek práce.

Základní příčina tohoto jevu je v nutnosti koordinace větších skupin lidí na větších projektech. Tato koordinace znamená zvýšenou nutnost komunikace a méně času zaměstnanců na vývoj. Velké množství kódu také znesnadňuje možné úpravy a opravy chyb.

Pro ilustraci uvádím v tabulce 2-1 přehled produktivity práce z různých druhů projektů o různé velikosti. Tabulka je převzata z (McConnell, 2006).

| Druh softwaru | Projekt 10 000 LOC | Projekt 100 000 LOC | Projekt 250 000 LOC |
|---|---|---|---|
| **Avionika** | 100 – 1 000 **(200)** | 20 – 300 **(50)** | 20 – 200 **(40)** |
| **Obchodní systémy** | 800 – 18 000 **(3 000)** | 200 – 7000 **(600)** | 100 – 5 000 **(500)** |
| **Velení a řízení** | 200 – 3 000 **(500)** | 50 – 600 **(100)** | 40 – 500 **(80)** |
| **Embedded systémy** | 100 – 2 000 **(300)** | 30 – 500 **(70)** | 20 – 400 **(60)** |
| **Internetové systémy** | 600 – 10 000 **(1 500)** | 100 – 2 000 **(300)** | 100 – 1 500 **(200)** |
| **Intranetové systémy** | 1 500 – 18 000 **(3 000)** | 300 – 7 000 **(800)** | 200 – 5 000 **(600)** |

| Druh softwaru | Projekt 10 000 LOC | Projekt 100 000 LOC | Projekt 250 000 LOC |
|---|---|---|---|
| **Mikrokód** | 100 – 800 **(200)** | 20 – 200 **(40)** | 20 – 100 **(30)** |
| **Řízení procesu** | 500 – 5 000 **(1 000)** | 300 – 7 000 **(300)** | 80 – 900 **(200)** |
| **Real-time aplikace** | 100 – 1 500 **(200)** | 20 – 200 **(50)** | 20 – 300 **(40)** |
| **Vědecké systémy** | 500 – 7 500 **(1 000)** | 100 – 1 500 **(300)** | 80 – 1 000 **(200)** |
| **Krabicový systém** | 400 – 5 000 **(1 000)** | 100 – 100 **(200)** | 70 – 800 **(200)** |
| **Systémový software a ovladače** | 200 – 5 000 **(600)** | 50 – 1 000 **(100)** | 40 – 800 **(90)** |
| **Telekomunikace** | 200 – 3 000 **(600)** | 50 - 600 **(100)** | 40 – 500 **(90)** |

**Tabulka 2-1 Produktivita u různých druhů software. Tabulka převzata z (McConnell, 2006).**

Pokud pro přepočet velikosti projektu na práci používáme jednoduchý koeficient (jako například řádky kódu napsané zaměstnancem za měsíc práce), je nutné tento koeficient počítat z podobně velikých projektů jako je ten odhadovaný.

### 2.1.2 Podhodnocování či nadhodnocování odhadů

Je zřejmé, že nejlepší odhad je přesný odhad. Dosáhnout takového stavu je však složité a většinou se stane, že je odhad vůči realitě podhodnocen nebo nadhodnocen. Pokud už tedy v odhadu chybujeme, je lepší odhad nadhodnocený nebo podhodnocený?

Nejprve se zaměřím na důsledky nadhodnocení. U takových projektů často vchází v platnost tzv. Parkinsonův zákon. Ten říká, že se vždy najde dostatek práce, aby byl využit všechen naplánovaný čas. Pokud tedy dáme vývojáři 3 dny na úkol, který lze zvládnout za 2 dny, jistě si na zbylý čas najde nějakou práci. Další věcí týkající se nadhodnocených projektů je tzv. Studentský syndrom. Ten nám říká, že pokud budou mít vývojáři příliš mnoho času na dokončení úkolů, budou práci odkládat a intenzivně začnou pracovat až pozdě a projekt nedokončí včas. Parkinsonovu zákonu se asi vyhnout nemůžeme, ale na studentský syndrom je jistě účinným lékem důsledná kontrola plnění zadaných úkolů a aktivní řízení projektu.

Důsledkům nadhodnocení projektu se někdy chtějí manažeři vyhnout tím, že projekt vědomě podhodnotí, aby mu dodali jistý nádech naléhavosti a vyhnuli se tak Parkinsonovu

zákonu i Studentskému syndromu u vývojářů. Zda ovšem možné důsledky podhodnocení projektu nejsou mnohem hrozivější, proberu v následujícím odstavci.

Podhodnocením odhadu si podkopáváme možnost efektivního plánování projektu a plánování návazností na ostatní projekty. Můžeme naplánovat menší tým, než je ve skutečnosti potřeba a ve chvíli, kdy se podhodnocení odhalí, nemusí být již potřebné kapacity dostupné. Navíc jakmile se podhodnocený projekt dostane do zpoždění, je tým zaměstnán spoustou činností, které za normálního stavu dělat nemusí. To způsobí, že zpoždění může být mnohem větší než původní podhodnocení. Příkladem takovýchto činností může být například:

- přepracovávání odhadu pro určení data skutečného dokončení projektu
- průběžná vydání softwaru pro zákazníka, kvůli předem plánovaným prezentacím projektu atd.
- opravy problémů vyplývajících ze zjednodušených řešení vytvářených pod časovým tlakem

Na žádnou z těchto aktivit by v případě správně naplánovaného projektu vůbec nedošlo.

Pokud srovnám důsledky podhodnocení a nadhodnocení projektů, vychází mi, že nadhodnocené projekty se díky Parkinsonovu zákonu prodraží lineárně vzhledem k velikosti nadhodnocení a navýšení je navíc ohraničené (vyplní všechen dostupný čas, ale ne více). Zatímco projekty podhodnocené se prodraží více než lineárně a prodražení není ničím omezené. Toto je způsobeno dynamikou zpožděných projektů, která je zapříčiněna specifickými aktivitami popsanými výše.

Závěrem tedy je, že podhodnocení projektu je potenciálně mnohem nebezpečnější než jeho nadhodnocení a může způsobit mnohem větší škody.

## 2.2 Volba způsobu měření velikosti software

Způsob měření velikosti projektu je velmi důležitou součástí odhadování softwarových projektů. Jednotky pro měření velikosti software je možné rozdělit do dvou větších skupin. První skupinou jsou jednotky technické, které vyjadřují velikost pomocí nějakého technického parametru aplikace. Tento parametr lze většinou poměrně snadno zjistit z dokončené aplikace. Druhou skupinou jsou jednotky funkční, které vyjadřují velikost projektu pomocí jisté abstrakce.

Já se budu dále v textu podrobněji zabývat dvěma způsoby určení velikosti. Konkrétně se jedná o způsob asi nejstarší a do dneška nejpoužívanější, tedy počítání řádek kódu. Dále se zaměřím na způsob modernější a také hojně používaný, a to počítání funkčních bodů.

### 2.2.1 Technický způsob měření velikosti

Technické jednotky určují velikost projektů pomocí nějakého technického parametru. Tímto parametrem může být například počet řádků kódu, počet větších celků projektu, počet souborů, počet databázových tabulek a mnohé další.

Tento způsob měření velikosti je poměrně jednoduchý a nevyžaduje větší úsilí, poněvadž většina moderních nástrojů na vývoj aplikací má přímou podporu pro počítání některých těchto parametrů.

V této práci se podrobněji zaměřím na nejrozšířenějšího zástupce této třídy a měření velikosti vůbec, tzn. na počítání řádek kódu.

#### 2.2.1.1 Počítání řádek kódu

Určování velikosti softwaru pomocí počtu řádek kódu (SLOC – Source Lines Of Code nebo jen LOC – Lines Of Code) je i přes své nevýhody jistě nejrozšířenějším způsobem měření velikosti projektu.

Počítání řádek je velice jednoduchý způsob pro vyjádření velikosti aplikace. Většina moderních vývojových nástrojů nám zobrazuje přehled o počtu řádek kódu, máme tedy tuto velikost stále k dispozici bez jakéhokoli vynaloženého úsilí z naší strany. Aby však byly velikosti jednotlivých aplikací v rámci společnosti porovnatelné, je potřeba vydefinovat, co to vlastně takový řádek kódu je a výsledný standard pro počítání řádků striktně dodržovat. Je tedy nutné odpovědět na následující otázky:

- Počítat všechen kód nebo jen kód, který je obsažen ve výsledném software (release)? (započítáváme tedy také pomocný kód, kód simulovaných objektů, kód automatických testů… nebo jen kód výsledné aplikace)
- Započítávat kód pocházející z dřívějších verzí?
- Započítávat kód knihoven třetích stran?
- Započítávat prázdné řádky a komentáře anebo jen řádky s kódem?
- Započítávat rozhraní tříd?
- Započítávat deklarace dat?

- Jak započítat jeden logický řádek kódu rozdělený kvůli čitelnosti na několik řádků?

Neexistuje žádný obecně definovaný a přijímaný standard pro počítání řádek. Je tedy zcela na každém, jak je počítá. To je ovšem z mého pohledu jistá nevýhoda v případě, že bychom chtěli využívat data o projektech z jiných firem ze stejného odvětví, například pro účely odhadování projektů. Nemůžeme si být jisti, zda řádky počítají stejně jako my, a zda si proto do odhadování nezaneseme další zdroj nepřesností.

Výhodou tohoto způsobu počítání velikosti software je v první řadě jednoduchost získání těchto dat z minulých projektů a za druhé rozšířenost tohoto způsobu měření. Většina společností využívá tento způsob pro měření velikosti software a také většina nástrojů pro odhadování v závěru stanoví odhad počtem řádků kódu. Takže existuje velké množství dat ve spoustě firem, která je možno použít. I když použití těchto dat je omezeno problémem neexistujícího standardu počítání LOC zmíněným výše. Dalším problémem je to, že velikosti srovnatelných projektů napsaných v jiných programovacích jazycích se mohou velmi lišit. Navíc se v každé společnosti používá jiná konvence psaní kódu a každý programátor napíše stejnou funkčnost na jiný počet řádků.

Z nevýhod LOC zmíním to, že jednoduchý model pro určení efektivity "počet řádků kódu na měsíc práce zaměstnanců" nebude dobře fungovat na různě velké projekty z důvodu existence "nevýhody velikosti" u softwaru (kapitola 2.1.1) a také různé produktivity u různých typů software. Další nevýhodou je to, že odhadování velikosti pomocí řádek kódu je v dřívějších fázích projektu jistě velmi neintuitivní. Navíc LOC můžeme na rozdíl od zástupců funkčního způsobu měření velikosti vždy počítat až po ukončení projektu, do té doby je můžeme jen odhadovat.

Počítání pracnosti na základě LOC navíc nezohledňuje existenci moderních nástrojů na automatické generování kódu.

### 2.2.2 Funkční způsob měření velikosti

Funkční jednotky pro určování velikosti nevyjadřují určité technické měřítko, které reflektuje provedení konkrétního projektu. Snaží se naopak pomocí nějakého syntetického měřítka zohlednit velikost aplikace z funkčního hlediska.

Tento způsob měření velikosti by měl být nezávislý na konkrétním řešení projektu a programovacím jazyce, ve kterém je projekt vytvářen.

Dále se podrobněji podívám na poměrně rozšířeného zástupce této třídy, a to na počítání funkčních bodů.

#### 2.2.2.1 Počítání funkčních bodů

Hlavním zástupcem skupiny funkčních jednotek velikosti jsou funkční body (FP – Functional Points), někdy se také uvádí funkční celky.

Funkční body jsou syntetickým měřítkem velikosti projektu, které by mělo, na rozdíl od řádků kódu, vyjadřovat velikost aplikace z hlediska funkčnosti a ne z hlediska zpracování konkrétního řešení. Toto měřítko by mělo být nezávislé na programovacím jazyce a mělo by představovat kvalitativně lepší předpoklady pro srovnávání projektů než měřítka technická.

Způsob počítání funkčních bodů je standardizován a udržován mezinárodní organizací uživatelů funkčních bodů (IFPUG – International Function Point Users Group)[1].

Konkrétní počet funkčních bodů je dán počtem a složitostí všech následujících položek:

- **Externí vstupy** (EI – External Inputs) – Obrazovky, formuláře, dialogy nebo řídící signály, pomocí nichž koncový uživatel nebo jiný program mění, maže nebo přidává data programu. Obsahují veškeré vstupy, které mají unikátní formát nebo logiku zpracování.
- **Externí výstupy** (EO – External Outputs) – Obrazovky, reporty, grafy nebo řídící signály, které program generuje pro koncového uživatele nebo jiný program. Obsahují veškeré vstupy, které mají různé formáty nebo navzájem různou logiku zpracování.
- **Externí odkazy** (EQ – External Inquires) – Kombinace vstup/výstup, ve které vstup vyústí v okamžitý jednoduchý výstup. Pojem odpovídá přímému vyhledávání jednoduchých dat v databázi, obyčejně s použitím jednoduchého klíče. V moderních uživatelských rozhraních a webových aplikacích je ovšem hranice mezi výstupy a dotazy nejasná. Lze ale obecně říci, že dotazy získávají data přímo z databáze, dávají jim jen jednoduché formátování; zatímco výstupy mohou data zpracovávat, kombinovat a složitěji formátovat pro výstup.
- **Interní logické soubory** (ILF – Internal Logical Files) – Veškeré logické soubory dat koncového uživatele, které jsou plně řízeny programem. Logický soubor se může skládat z jednoho souboru nebo jedné tabulky v relační databázi.

[1] Oficiální stránky této organizace najdete na www.ifpug.org

- **Externí soubory rozhraní** (EIF – External Interface Files) – Soubory řízené jinými programy, s kterými váš projekt spolupracuje. Jde o každou logickou skupinu dat nebo řídících informací, které vstupují nebo vystupují z programu.

Tabulka 2-2 ukazuje, jak se počet všech těchto elementů přepočítá na počet neupravených funkčních bodů (UFP – Unadjusted Function Points). Jednotlivé elementy se rozdělí podle složitosti do tří skupin a jejich počty se následně vynásobí koeficientem z příslušného řádku a sloupce tabulky. Celkový součet těchto čísel nám pak dá počet neupravených funkčních bodů (UFP – Unadjusted Function Points).

| Entita | Malá složitost | Střední složitost | Velká složitost |
|---|---|---|---|
| Externí vstupy | 3 | 4 | 6 |
| Externí výstupy | 4 | 5 | 7 |
| Externí dotazy | 3 | 4 | 6 |
| Interní logické soubory | 4 | 10 | 10 |
| Externí soubory rozhraní | 5 | 7 | 15 |

**Tabulka 2-2 Tabulka s koeficienty příslušných kategorií a složitostí elementů pro výpočet UFP tak, jak je uvedena v (McConnell, 2006) a (Themis, 1997)**

Výsledný počet těchto neupravených funkčních bodů je následně nutné upravit pomocí koeficientu vypočítaného z charakteristiky projektu, která je vyjádřena pomocí následujících 14 faktorů, jak jsou uvedeny v (Longstreet, 2001):

1) Vyžaduje systém spolehlivé zálohování a zotavení?
2) Jsou vyžadovány datové komunikace?
3) Existuje distribuované zpracování?
4) Je výkonnost kritická?
5) Poběží systém v stávajícím intenzivně využívaném operačním prostředí?
6) Požaduje systém on-line vstup dat?
7) Je pro on-line vstup dat vyžadováno použití vstupní transakce přes více obrazovek nebo operací?
8) Jsou hlavní soubory opravovány on-line?

9) Jsou vstupy, výstupy, soubory a dotazy složité?
10) Je vnitřní zpracování složité?
11) Je kód navrhován s cílem znovupoužití?
12) Jsou konverze a instalace zahrnuty v návrhu?
13) Je systém navrhován pro násobné instalace u různých organizací?
14) Je aplikace navrhovaná tak, aby zajistila změny a snadné používání na straně uživatele?

Každý faktor je ohodnocen číslem 0 – 5 podle toho, jak vystihuje vlastnosti projektu. Stupnice hodnocení je následující:

- 0 = bez vlivu
- 1 = náhodný
- 2 = mírný
- 3 = průměrný
- 4 = významný
- 5 = podstatný

Výsledný koeficient vlivu systémových charakteristik (VAF) se vypočte podle následujícího vzorce:

$$VAF = 0{,}65 + \sum_{i=1}^{14} C_i/100$$

Kde $C_i$ je ohodnocení i-té charakteristiky.

Neupravené funkční body se pak převedou na upravené funkční body (AFP – Adjusted Function Points, nebo jen FP) pomocí následujícího vzorce:

$$FP = UFP * VAF$$

Pro demonstraci postupu je uveden v následující tabulce příklad výpočtu funkčních bodů na ilustračním projektu.

| Entita | Malá složitost | Střední složitost | Velká složitost |
|---|---|---|---|
| Externí vstupy | 3 x 3 = 9 | 4 x 4 = 16 | 2 x 6 = 12 |
| Externí výstupy | 5 x 4 = 20 | 2 x 5 = 10 | 3 x 7 = 21 |
| Externí dotazy | 2 x 3 = 6 | 3 x 4 = 12 | 1 x 6 = 6 |
| Interní logické soubory | 0 x 4 = 0 | 1 x 10 = 10 | 2 x 10 = 20 |
| Externí soubory rozhraní | 6 x 5 = 30 | 2 x 7 = 14 | 2 x 15 = 30 |
| **Počet neupravených funkčních bodů** | | | **216** |
| **Koeficient pro přepočet na upravené body** | | | **1,14** |
| **Výsledný počet upravených funkčních bodů** | | | **246** |

**Tabulka 2-3 Tabulka demonstrující výpočet upravených funkčních bodů projektu**

Projekt uvedený v příkladu má tedy velikost 246 upravených funkčních bodů (nebo jen funkčních bodů).

Počítání funkčních bodů bylo původně určeno především pouze pro určování velikosti databázových aplikací, ale díky činnosti sdružení IFPUG[2] byla pravidla pro počítání postupně upravována a dnešní verze je již aplikovatelná na všechny druhy softwaru.

Podrobnější postup počítání funkčních bodů najdete v kapitole 6.5.2 a v (Longstreet, 2001).

#### 2.2.2.2 Počítání Use Case bodů

Dalším zástupcem skupiny funkčního způsobu měření velikosti software je počítání use case bodů (UCP – Use Case Points). Use case body jsou syntetickým měřítkem velikosti projektu, analogickým k funkčním bodům. Místo funkčních celků se zde ale počítají diagramy use case a kalkuluje se jejich složitost.

Rovnice na výpočet UCP je následující:

$$UCP = UUCP * TCF$$

Kde

- UUCP (Unadjusted Function Points) znamená neupravené use case body
- TCF (Technical Complexity Factor) znamená technický koeficient komplexity

[2] Mezinárodní skupina uživatelů funkčních bodů (www.ifpug.cz)

Pro výpočet UUCP budeme potřebovat nejprve spočítat hodnotu UUCW (Unadjusted Use Case Weight) a UAW (Unadjusted Actor Weight).

| UUCW – Unadjusted Use Case Weight | | |
|---|---|---|
| **Typ Use Case** | **Popis** | **Váha** |
| Simple | Jednoduché uživatelské rozhraní. Pracuje pouze s jednou databázovou entitou. Jeho úspěšný scénář má tři kroky, nebo méně. Jeho implementace se skládá z méně než pěti tříd. | 5 |
| Average | Složitější uživatelské rozhraní. Pracuje se dvěma nebo více databázovými entitami. Uspěný scénář má mezi čtyřmi a sedmi kroky. Implementace zahrnuje pět až deset tříd. | 10 |
| Complex | Komplexní uživatelské rozhraní, nebo zpracování. Pracuje se třemi nebo více databázovými entitami. Úspěšný scénář je o více než sedmi krocích. Implementace se skládá z více než deseti tříd. | 15 |

**Tabulka 2-4 Tabulka pro výpočet UUCW, převzata z (Cohn, 2005)**

**UUCW** tedy spočteme tak, že use case diagramy rozdělíme do tříd podle složitosti a jejich počty vynásobíme koeficienty z tabulky. Výsledná čísla pak sečteme a dostaneme počet neupravených use case vah.

| UAW – Unadjusted Actor Weight | | |
|---|---|---|
| **Typ Aktéra** | **Popis** | **Váha** |
| Simple | Aktér představuje jiný systém. Je definováno aplikační rozhraní. | 1 |
| Average | Aktér představuje jiný systém. Interakce je uskutečněna prostřednictvím protokolů, jako jsou TCP/IP. | 2 |
| Complex | Aktér je člověk, který s aplikací interaguje pomocí vnějšího rozhraní. | 3 |

**Tabulka 2-5 Tabulka pro výpočet UAW jak ji uvádí (Cohn, 2005)**

**UAW** spočteme tak, že aktéry rozdělíme do tříd podle složitosti a jejich počty vynásobíme koeficienty z tabulky. Výsledná čísla sečteme a dostaneme počet neupravených vah aktérů.

Výsledný počet neupravených use case bodů pak získáme, když sečteme UUCW a UAW. Tedy platí následující vztah:

$$UUCP = UUCW + UAW$$

TCF následně vypočteme ohodnocením charakteristik projektu uvedených v tabulce 2-6.

| TCF – Technical Complexity Factor | | |
|---|---|---|
| **Technický faktor** | **Popis** | **Váha** |
| T1 | Distribuovaný systém | 2 |
| T2 | Výkon | 1 |
| T3 | Uživatelská jednoduchost | 1 |
| T4 | Komplexní vnitřní zpracování | 1 |
| T5 | Znuvupoužitelnost | 1 |
| T6 | Snadná instalace | 0.5 |
| T7 | Snadné používání | 0.5 |
| T8 | Portovatelnost | 2 |
| T9 | Snadná upravitelnost | 1 |
| T10 | Souběžný běh | 1 |
| T11 | Silné zabezpečení | 1 |
| T12 | Povinná speciální školení uživatelů | 1 |
| T13 | Poskytuje přímý přístup k třetím stranám | 1 |

**Tabulka 2-6 Tabulka pro výpočet faktoru pro TCF převzata z (Cohn, 2005)**

Každá charakteristika je ohodnocena číslem 0 – 5 podle toho, jak vystihuje vlastnosti projektu. Stupnice hodnocení je následující:

- 0 = bez vlivu
- 1 = náhodný
- 2 = mírný

- 3 = průměrný
- 4 = významný
- 5 = podstatný

Výsledný faktor spočteme sečtením jednotlivých součinů vah a ohodnocením jednotlivých charakteristik. Faktor TCF pak spočítáme podle následujícího vzorce:

$$TCF = 0,6 + (0,01 * \text{Výsledný faktor})$$

Výsledná rovnice pro výpočet velikosti projektu v use case bodech (tedy počet UCF) je následující:

$$UCP = UUCP * TCF$$

Podrobnější postup pro počítání use case bodů najdete v (Cohn, 2005).

### 2.2.3 Převoditelnost různých vyjádření velikosti projektů

Pro převod mezi velikostmi projektů vyjádřenými v odlišných jednotkách existují tabulky vytvořené z dat o projektech z různých odvětví. Tyto obecné převodní tabulky jsou ovšem nepřesné. Pokud tedy potřebujete ve vaší společnosti převádět například počty funkčních bodů na počty LOC, dosáhnete vyšší přesnosti, pokud si nasbíráte vlastní data a pomocí nich si spočtete vlastní převodní koeficienty. Je to způsobeno tím, že projekty v rámci jedné společnosti jsou velmi podobné a navíc styl kódování, programovací jazyk a postup počítání řádků (tedy vlastnosti vývoje nejvíce ovlivňující počet LOC) jsou shodné.

Následuje příklad obecné převodní tabulky (tabulka 2-7) pro převod počtu funkčních bodů na počet LOC pro různé programovací jazyky.

| **Jazyk** | **Minimum (Střed mínus standardní odchylka)** | **Střed (nejčastější hodnota)** | **Maximum (Střed plus standardní odchylka)** |
|---|---|---|---|
| Ada 83 | 45 | 80 | 125 |
| Ada 95 | 30 | 50 | 70 |
| C | 60 | 128 | 170 |

| Jazyk | Minimum (Střed mínus standardní odchylka) | Střed (nejčastější hodnota) | Maximum (Střed plus standardní odchylka) |
|---|---|---|---|
| C# | 40 | 55 | 80 |
| C++ | 40 | 55 | 140 |
| Cobol | 65 | 107 | 150 |
| Fortran 77 | 65 | 107 | 160 |
| Fortran 90 | 45 | 80 | 125 |
| Fortran 95 | 30 | 71 | 100 |
| Java | 40 | 55 | 80 |
| Macro Assembly | 130 | 213 | 300 |
| Perl | 10 | 20 | 30 |
| Pascal | 65 | 107 | 160 |
| Smalltalk | 10 | 20 | 40 |
| SQL | 7 | 13 | 15 |
| MS Visual Basic | 15 | 32 | 41 |

**Tabulka 2-7 Počet LOC na jeden upravený funkční bod pro několik programovacích jazyků. Tabulka je převzata z (McConnell, 2006).**

Pro demonstraci použiji ilustrační program, který jsem využil pro počítání FP. Budu předpokládat, že je napsán v programovacím jazyce Java. Počet 246 FP převedu podle příslušného řádku tabulky na odhad velikosti 9 840 LOC až 19 680 LOC s očekávanou hodnotou 13 530 LOC.

Je vidět, že výsledný interval velikosti v LOC je poměrně široký. Jak jsem se již zmínil, zpřesnění je možné dosáhnout pomocí použití vlastních historických dat.

# 3 Metody odhadování projektů

V této kapitole uvedu několik metod používaných pro odhadování projektů, které jsem vybral z metod uvedených v (Coombs, 2003), (Themis, 1997) a (McConnell, 2006). Z uvedených metod jsem také čerpal při výběru metod pro použití v návrhu metodiky, která je hlavní součástí této práce. Mojí snahou je uvést u každé metody pouze stručný přehled jejích vlastností a postupu jejího použití. Pro podrobnější informace musí čtenář sáhnout po odborné literatuře. Některé tituly, ze kterých je možno čerpat, jsou uvedeny v seznamu literatury použité při tvorbě této práce. Seznam je uveden na konci práce.

Metody jsou rozděleny podle toho, zda pomocí nich odhadujeme velikost projektu nebo zda přímo odhadujeme potřebnou práci. Dále je zde uveden způsob, jak lze převést velikost na práci a také stručný popis těch historických dat, která se vyplatí sbírat.

U každé metody rovněž uvádím jednoduché zhodnocení predikčních schopností a možnosti použití s ohledem na fázi rozpracovanosti projektu a velikost[3] projektu.

## 3.1 Metody přímého odhadování práce

Pomocí metod přímého odhadování práce rovnou odhadujeme práci, kterou je potřeba vynaložit na dokončení projektu. Nepoužíváme tedy žádného zástupce a není potřeba převádět odhad velikosti na práci. Odhadování pomocí těchto metod není závislé na historických datech, která by tyto převody umožňovala.

V této skupině odhadů převládají expertní úsudky či různé principy analogie, při kterých využíváme historická data.

Nyní uvedu několik metod spadajících do této skupiny přímého odhadování práce.

### 3.1.1 Individuální expertní úsudek

Individuální expertní úsudek je jistě nejčastější metodou pro odhadování práce na projektech. Je to také ovšem metoda, která je nejvíce náchylná k nepřesnostem.

---

[3] Rozdělení na třídy podle velikosti software je následující:
***Malý projekt (M)*** – projekty s pěti nebo méně technickými pracovníky
***Střední projekt (S)*** – projekty, které trvají od 3 do 12 měsíců, a pracuje na nich od 5 do 25 technických pracovníků
***Velký projekt (V)*** – projekt, který má více jak 25 technických pracovníků a doba trvání je od 6 do 12 měsíců nebo delší

Odhad touto metodou většinou tvoří odborníci na technologii a způsoby vývoje použité v odhadovaném projektu. Odhadují buď práci na celý projekt najednou, ale častější je odhadování jednotlivých vlastností projektu a následná kalkulace celkového odhadu. Je to jistý způsob odhadu metodou zdola nahoru.

Tabulka 3-1 ukazuje stručnou charakteristiku této metody.

| **Použitelnost odhadu pomocí individuálního expertního úsudku** | |
|---|---|
| Velikost Projektu | M S V |
| Fáze projektu | Počáteční až pozdní |
| Přesnost odhadu | Střední - Vysoká |

**Tabulka 3-1 Charakteristika odhadu pomocí individuálního expertního úsudku**

Pro minimalizaci rizika nepřesnosti odhadu existuje mnoho postupů. Několik z nich si podrobněji probereme. Pro zlepšení odhadů je můžeme použít jednotlivě nebo je možné je kombinovat.

### Použití intervalového odhadu

Pokud budeme požadovat odhady vyjádřené jedním číslem, vystavujeme se většinou riziku, že odhadovatelé uvedou odhad ne v očekávaném případě, ale spíše odhad v nejlepším nebo nejhorším případě. Naproti tomu pokud budeme chtít odhad v nejlepším a nejhorším případě (tedy jistou formu intervalu, proto intervalový odhad) jistým způsobem donutíme odhadovatele zamyslet se nad všemi riziky ovlivňujícími ten nejhorší případ. Tímto způsobem jistě přispějeme ke zpřesnění odhadu.

Intervalové odhady jsou nakonec výhodné i pro tvůrce odhadů, protože již po nich nechceme jedno číslo, ale jistý interval, jehož šířka by měla vyjadřovat míru nejistoty ve specifikaci projektu. Managementu pak výsledný intervalový odhad pomůže na tuto míru nejistoty přesněji nahlédnout.

Odhad se může ve výsledku prezentovat jako interval, nebo může být prezentována vypočítaná očekávaná hodnota (např. průměr z obou hodnot).

### Technika tří bodů

Technika tří bodů je postup, jak z vícebodového odhadu spočítat odhad jednobodový. Tato technika předpokládá, že vytvoříme tříbodový odhad projektu. Každý bod představuje

odhad projektu v různých situacích. Konkrétně v nejlepším případě, nejhorším případě a nejpravděpodobnějším případě. Výsledný odhad se poté spočítá podle následujícího vztahu:

Výsledný odhad = (Nejlepší + (4 * Nejpravděpodobnější) + Nejhorší) / 6

**Srovnání odhadů se skutečností**

Srovnávání odhadů se skutečností je velmi důležité pro uvědomění si kvalit odhadů, které vytváříte, a pro případné přizpůsobení odhadování, pokud se odhady například jednostranně vymykají skutečnosti.

Je to také dobrý způsob pro stanovení průměrné chyby vytvářených odhadů. Tuto chybu pak můžeme prezentovat společně s odhadem a dále s ní počítat například při rozpočtování projektu.

### 3.1.2 Skupinový expertní odhad

Skupinový expertní odhad je výhodnější pro rané fáze projektů, kdy se v nich vyskytují větší neznámé, na které se pohled různých členů týmu může výrazně lišit.

Tabulka 3-2 ukazuje stručnou charakteristiku této metody.

| **Použitelnost odhadu pomocí skupinového expertního úsudku** | |
|---|---|
| Velikost Projektu | S V |
| Fáze projektu | Počáteční až střední |
| Přesnost odhadu | Střední |

**Tabulka 3-2 Charakteristika odhadu pomocí skupinového expertního úsudku**

Přidaná hodnota jednoduché metody skupinového odhadu spočívá ve vzájemné konzultaci úsudků jednotlivých členů odhadovacího týmu, směřující k všeobecné akceptaci výsledného odhadu.

Postup této metody lze shrnout do následujících tří bodů:

- Každý člen týmu vytvoří individuální odhad. Tyto odhady se poté vzájemně porovnají a rozdíly se diskutují.
- Diskuse nad odhady by měla být důsledná a měla by se zaměřit na odhalení zdrojů rozdílů v jednotlivých odhadech a jejich eliminaci.

- Výsledný odhad by měl získat všeobecnou podporu v týmu odhalovatelů.

Složení týmu by mělo být různorodé, je tedy dobré, aby členové týmu svými znalostmi rovnoměrně pokrývali požadavky projektu. Počet členů týmu by měl být v nejlepším případě od 3 do 5.

### 3.1.3 Technika Wideband Delphi

Technika Wideband Delphi, jak ji uvádí (McConnell, 2006), je strukturovanou metodou skupinového úsudku. Její původní verze byla vyvinuta ve společnosti Rand Corporation v první polovině 20. století pro předpovídání trendů v technologii.

Tabulka 3-3 ukazuje stručnou charakteristiku této metody.

| **Použitelnost odhadu pomocí metody Delphi** | |
|---|---|
| Velikost Projektu | S V |
| Fáze projektu | Počáteční |
| Přesnost odhadu | Střední |

**Tabulka 3-3 Charakteristika odhadu pomocí Wideband Delphi**

Metoda se liší od jednoduchého skupinového odhadu tím, že je zde přesně specifikovaný postup při snaze o konvergenci jednotlivých odhadů k výslednému odhadu. Tvorba odhadu probíhá následovně v těchto několika krocích:

1) Koordinátor předloží každému odhalovateli specifikaci a formu odhadu.
2) Odhadovatelé připraví své individuální odhady.
3) Na skupinovém setkání se proberou problémy úzce spojené s předmětem odhadování.
4) Odhadovatelé dají koordinátorovi své individuální odhady.
5) Koordinátor předloží odhalovatelům souhrn všech odhadů, aby měli srovnání s tím vlastním.
6) Skupinově se rozeberou odchylky v jednotlivých odhadech.
7) Proběhne anonymní hlasování o přijmutí průměrného odhadu. Pokud kdokoli hlasuje proti, přejde se k bodu 3.
8) Finální odhad metodou Delphi je vytvořen.

Jednotlivé kroky je výhodnější provádět při skupinovém setkání, ale je možné je provést například i formou e-mailové konference.

### 3.1.4 Jednoduchá analogie

V této metodě využijeme pro odhadnutí projektu znalost práce spotřebované na nějakém předchozím projektu.

Tabulka 3-4 ukazuje stručnou charakteristiku této metody.

| Použitelnost odhadu pomocí jednoduché analogie | |
|---|---|
| Velikost Projektu | M S V |
| Fáze projektu | Počáteční |
| Přesnost odhadu | Střední - Vysoká |

**Tabulka 3-4 Charakteristika odhadu pomocí jednoduché analogie**

Základní metodou při odhadování pomocí analogie je tedy srovnání odhadovaného projektu s podobným projektem z minulosti, u kterého máme přehled o požadované práci. Po nalezení daného analogického projektu, můžeme ještě odhadnout procentuální poměr mezi velikostmi projektu nového a starého a v tomto poměru upravit i požadovanou práci.

Jestliže jsme schopni mezi minulými projekty nalézt dostatečně podobný projekt, dává tato metoda poměrně dobré výsledky.

### 3.1.5 Strukturovaná analogie

V této metodě je více rozvedena jednoduchá analogie. V tabulce 3-5 je stručná charakteristika metody.

| Použitelnost odhadu pomocí strukturované analogie | |
|---|---|
| Velikost Projektu | M S V |
| Fáze projektu | Počáteční až pozdní |
| Přesnost odhadu | Vysoká |

**Tabulka 3-5 Charakteristika odhadu pomocí strukturované analogie**

Princip zůstává stejný, ale porovnávání projektu je propracovanější. V rámci metody nejprve zvolíme nějaký analogický projekt a poté porovnáváme počty spočitatelných

elementů tohoto projektu a toho odhadovaného. Můžeme porovnávat například počty webových stránek, databázových tabulek, rozhraní atp.

Výsledný odhad se pak vypočte z poměrů těchto elementů a velikosti vybraného projektu.

## 3.2 Metody odhadování velikosti projektu

Pomocí metod uvedených v této kapitole odhadujeme velikost projektu vyjádřenou pomocí některého ze způsobů měření velikosti projektů. Výsledkem tedy nebude potřebná práce na projekt, ale například velikost v řádcích kódu.

Převedením odhadu velikosti na odhad práce se poté zabývám na konci kapitoly. Uvádím i několik typů pro sběr historických dat z projektů.

Dále je uvedeno několik metod spadajících do této skupiny odhadování velikosti. U každé metody je mimo jiné uvedeno, pro jaký způsob měření velikosti projektu jsou použitelné.

### 3.2.1 Odhadování FP pomocí extrapolace

Tato metoda je určena pro odhadování velikosti ve funkčních bodech a předpokládá závislosti mezi entitami pro výpočet funkčních bodů. Takže v situaci, kdy nejsme schopni spočítat všechny komponenty z důvodu omezené znalosti řešení, je možné spočítat jen některé komponenty a ostatní dopočítat pomocí závislostí.

Tabulka 3-6 ukazuje stručnou charakteristiku této metody.

| **Použitelnost odhadu pomocí extrapolace** | |
|---|---|
| Velikost Projektu | S V |
| Fáze projektu | Počáteční |
| Přesnost odhadu | Vysoká |
| Vhodný způsob měření velikosti | Funkční body |

**Tabulka 3-6 Charakteristika odhadu pomocí extrapolace**

V (Meli, a další, 1999) a (McConnell, 2006) existují již vyjádřené závislosti získané z obecných dat o softwarových projektech. Jsou to například tyto:

- **Tichenor ILF Model:**

$$Odh.UFP = \#ILF * 11{,}01 \ \ pro\ dávkové\ systémy$$

$$Odh.UFP = \#ILF * 14.93 \ \ pro\ transakční\ systémy$$

- **CNV AG:**

$$Odh.UFP = 7.3 * \#EO + 56$$

- **NESMA:**

$$Odh.UFP = 35 * \#ILF + 15 * \#EIF$$

- **ISBSG Benchmark:**

$$Odh.UFP = \frac{(7{,}4 * \#ILF)}{22} * 100$$

Tyto vztahy jsou vyjádřeny pomocí obecných dat. V případě dostatku vlastních historických dat je možné vypočítat vlastní koeficienty.

### 3.2.2 Předdefinovaná řešení

Odhad pomocí předdefinovaných řešení je založen na databázi ohodnocených řešení použitelných v projektech. Tato řešení se při odhadování přiřazují jednotlivým úkolům nebo celému projektu.

Tabulka 3-7 ukazuje stručnou charakteristiku metody.

| **Použitelnost odhadu pomocí předdefinovaných řešení** | |
|---|---|
| Velikost Projektu | M S V |
| Fáze projektu | Počáteční až pozdní |
| Přesnost odhadu | Vysoká |
| Vhodný způsob měření velikosti | Všechny |

**Tabulka 3-7 Charakteristika odhadu pomocí předdefinovaných řešení**

Metoda je jistou modifikací odhadování pomocí analogie s tím, že neodhadujeme pomocí srovnání s minulými projekty, ale rozdělíme projekt na předefinovaná řešení, která už máme odhadnutá. Je tedy nutné mít obsáhlou databázi těchto řešení, aby byla metoda použitelná na širší spektrum projektů.

Obvykle je tato metoda použita v kombinaci s ostatními, kdy se s předdefinovaným řešením ztotožní jen několik úkolů v projektu a ostatní se odhadnou jiným způsobem.

### 3.2.3 Odhadování pomocí průměrné komplexity

Tato metoda, uvedena v (Themis, 1997) předpokládá to, že v projektech se vyskytují komponenty pro výpočet funkčních bodů s podobnou průměrnou složitostí. Tedy pokud vypočteme průměrnou složitost komponent, můžeme pomocí této metody odhadovat počet funkčních bodů, aniž bychom znali detailní vlastnosti komponent.

Tabulka 3-8 ukazuje stručnou charakteristiku této metody.

| **Použitelnost odhadu pomocí průměrné komplexity** | |
|---|---|
| Velikost Projektu | S V |
| Fáze projektu | Počáteční |
| Přesnost odhadu | Střední - Vysoká |
| Vhodný způsob měření velikosti | Funkční body, Use Case body |

**Tabulka 3-8 Charakteristika odhadu pomocí průměrné komplexity**

Aplikaci této metody je vhodné pokud jsme schopni spočítat jednotlivé komponenty, ale nejsme schopni určit jejich složitost.

Příklad takové průměrné složitosti je uveden v tabulce 3-9, která čerpá z ISBSG benchmarku vývojových projektů.

| | **Průměrný počet UFP** |
|---|---|
| **ILF** | 7,4 |
| **EIF** | 5,5 |
| **EI** | 4,3 |
| **EO** | 5,4 |
| **EQ** | 3,8 |

**Tabulka 3-9 Průměrné složitosti komponent podle ISBSG benchmark**

S využitím této tabulky počet neupravených funkčních bodů (Unadjusted Function Points - UFP) odhadneme následujícím vztahem:

$$Odh.UFP = \#EI * 4{,}3 + \#EO * 5{,}4 + \#EQ * 3{,}8 + \#ILF * 7{,}4 * \#EIF * 3{,}8$$

Koeficienty v tabulce 3-9 jsou vypočítané z obecných dat. Při dostatku vlastních historických dat mohou být tyto koeficienty nahrazeny koeficienty vlastními, vypočítanými z databáze.

Metodu lze analogicky použít pro odhad velikosti projektu v use case bodech. Místo průměrné složitosti elementů ve funkčních bodech se kalkuluje s průměrnou složitostí use case diagramů a aktérů.

### 3.2.4 Early Function Points

Tato metoda, uvedena v (Meli, 1997), je zaměřena na to, že v počátečních fázích projektu je detail rozpracování jednotlivých úkolů různý. Snaží se tedy umožnit jít při odhadování u jednotlivých úkolů vždy na nejnižší úrovně rozpracování.

Tabulka 3-10 ukazuje stručnou charakteristiku metody.

| **Použitelnost odhadu pomocí early function points** | |
|---|---|
| Velikost Projektu | M S V |
| Fáze projektu | Počáteční |
| Přesnost odhadu | Střední - Vysoká |
| Vhodný způsob měření velikosti | Funkční body |

**Tabulka 3-10 Charakteristika odhadu pomocí Early Function Points**

Pro odhadování definuje metoda jistou hierarchii, abychom mohli mapovat funkcionalitu u těch nejméně i nejvíce rozpracovaných úkolů. Tato hierarchie je následující:

- funkční primitiva
- mikrofunkce
- funkce
- makrofunkce

Metoda je podrobněji podepsaná v (Meli, a další, 1999) a (Meli, 1997).

### 3.2.5 Backfiring

Tato metoda používá pro odhadování počtu funkčních bodů převod z velikosti v LOC. Metoda je založená na výzkumu Caperse Jonese. Tabulka obsažená v (Themis, 1997) přináší přehled úrovní programovacích jazyků. Čím vyšší úroveň, tím méně příkazů je potřeba na jeden funkční bod.

Tabulka 3-11 ukazuje stručnou charakteristiku této metody.

| **Použitelnost odhadu pomocí metody Backfiring** | |
|---|---|
| Velikost Projektu | M S V |
| Fáze projektu | Počáteční - Pozdní |
| Přesnost odhadu | Střední |
| Vhodný způsob měření velikosti | Funkční body, Use Case body |

**Tabulka 3-11 Charakteristika odhadu pomocí metody Backfiring**

### 3.2.6 Use Case mapping

Touto metodou odhadujeme velikost ve funkčních bodech pomocí use case diagramů. Počet use case diagramů se podle počtu a jejich složitosti převede na počet funkčních bodů. Tato metoda potřebuje k nastavení delší historii programů, ze které se bude moci určit, zda existuje vztah mezi diagramy a počtem funkčních bodů a budeme z ní moci určit koeficienty pro převod.

## 3.3 Převod odhadu velikosti na odhad práce

Pokud provedeme odhad velikosti projektu nějakou z předchozích metod, jako výsledek dostaneme velikost projektu. Jsme tedy postaveni před problém, jak tuto odhadovanou velikost převést na odhad potřebné práce zaměstnanců do dokončení projektu.

Práce zaměstnanců se většinou udává v jednotkách značících práci jednoho člověka po dobu nějakého časového úseku. Je to například MD (Man-Day – jeden den práce zaměstnance), MM (Man-Month – jeden měsíc práce zaměstnance) a MY (Man-Year – rok práce zaměstnance). Pro převod jednotek velikosti projektu na jednotky práce zaměstnance je potřeba mít k dispozici nějaký koeficient. Tímto koeficientem může být produktivita zaměstnanců.

Produktivitou myslíme to, že zaměstnanci jsou například v průměru schopni napsat 500 řádek kódu za jeden den. Pokud tedy chceme převést odhad, že projekt bude mít 20 000 řádků kódu, na odhad práce, provedeme to následujícím výpočtem:

$$20\,000 / 500 = 40$$

Výsledkem tedy je, že potřebná práce na dokončení projektu je 40 MD. Jestliže označíme písmenem P produktivitu (počet řádek kódu napsaných jedním zaměstnancem za jeden den), písmenem V označíme velikost projektu (v řádcích kódu) a písmenem R označíme potřebnou práci (v MD) dostáváme následující vztah pro převod velikosti na práci:

$$R = V / P$$

Tento vztah je obecně použitelný pro jakýkoliv způsob měření velikosti projektu. Při jeho používání je ovšem nutné dát si pozor na několik věcí. Prvním z těchto problémů je problém velikosti software popsaný v úvodu této práce. Je tedy vhodné pro určení produktivity a přepočítávání velikosti určitého projektu používat data z podobně velkých projektů. Jinak by mohlo dojít ke zkreslení výsledného odhadu. Dalším poznatkem pro určování produktivity je to, že bychom měli používat data z projektů s podobným typem a hlavně také z projektů vyvíjených nejlépe ve stejném programovacím jazyce. Poslední bod je zejména při převodu z řádků kódu velmi důležitý.

Data pro výpočet produktivity mohou být z několika zdrojů. Mohou to být obecná data ze stejného odvětví, jakým se zabývá vaše společnost. Dále to mohou být například data o projektech vyvíjených ve stejném programovacím jazyce napříč odvětvími atp.

Osobně bych ovšem doporučil, pokud je to jen trochu možné, použít data vlastní. Použitím vlastních dat se razantně zvýší přesnost odhadů. Sběr historických dat se navíc jistě nevyužije jen na odhadování, ale například i na controlling projektů a jiné důležité činnosti. Více o sběru historických dat a o tom, jaká data sbírat najdete v následující kapitole.

### 3.3.1 Sběr historických dat

Sběr historických dat je velmi důležitá součást odhadování projektů. Nasbíraná data se využívají například pro převod velikosti na práci. Nejsou navíc užitečná jen pro odhadování, ale i pro jiné účely IT managementu.

Historická data můžeme získat z následujících tří zdrojů:

- historická data z odvětví
- historická data ze společnosti
- historická data z projektu

Historická data z odvětví, to jsou data, která pochází z projektů firem ze stejného odvětví jako vaše firma. Tato data se dají použít v případě, že nemáte data vlastní. Jejich přesnost je ovšem omezená, poněvadž i produktivita firem ze stejného odvětví se může podstatně lišit. Toto můžeme ilustrovat údaji z tabulky 2-1, kde je vidět že data z různých společností se mohou lišit až o desetinásobek.

Mnohem přesnější jsou data nasbíraná přímo z projektů ve vaší společnosti. Jejich přesnost je založena na tom, že na rozdíl od dat z jiných firem, vaše data reflektují specifika vývoje ve vaší společnosti a odráží se v nich řada faktorů, které se mohou napříč společnostmi lišit. Proto, pokud je to jen trochu možné, sbírejte vlastní data a použití těch obecných se snažte vyhnout. Pro zpřesnění těchto dat je možné nadefinovat a využít jistou metriku podobnosti projektu. Pro odhadování bychom následně používali pouze data z dostatečně podobných projektů.

Posledním zdrojem dat jsou data přímo z vyvíjeného projektu. Tato data jsou přímo z projektu, na kterém pracujeme, tedy data o produktivitě z předchozí fáze projektu anebo to mohou být také data například z předchozího release projektu. Takto získaná data jsou pro použití v projektu nejpoužitelnější. Měla by generovat nejpřesnější výsledky, protože jsou to data, která reflektují přímo faktory ovlivňující konkrétní projekt.

#### 3.3.1.1 Jaká data sbírat

Pokud se sběrem dat budete ve vaší společnosti začínat, potom jsou pro vás nejdůležitější tyto skupiny dat:

- velikost projektu (velikost projektu udaná v nějaké metrice)
- práce (práce potřebná na dokončení projektu)
- čas (časové vymezení jednotlivých fází)
- chyby (počet chyb na projektu, jejich závažnost a místo vzniku)

Tyto hlavní skupiny dat by měly pokrýt nejen potřeby odhadování, ale také potřeby sledování efektivity vývoje, chybovosti projektů a také reportingu. Podrobněji rozeberu, k jakým účelům slouží jednotlivá data.

Velikost projektu a odvedená práce nám dává možnost použití našich historických dat pro výpočet koeficientu pro převod velikosti na odhad práce. Tento převod je detailněji popsán dříve v této kapitole.

Pomocí dat o časovém vymezení projektů můžeme získat přehled o časové náročnosti budoucích projektů. A počty chyb z projektů nám dají možnost odhadovat chybovost a náročnost na podporu.

Více o sbírání historických dat se můžete dozvědět z (Rico, 2004) a nebo (Grady, et al., 1987).

# 4 Analýza současného stavu tvorby odhadů softwarových projektů v Komerční bance

V současné době neexistuje v KB žádná standardizovaná metodika pro odhadování pracnosti softwarových projektů. Odhady se tvoří vždy na konci jednotlivých fází projektu v projektové metodice. Odhady slouží pro kapacitní plánování, jako podklad pro rozhodování o projektu při přechodu mezi jednotlivými fázemi a pro plánování projektu.

Dále v textu uvádím popis tvorby odhadů v jednotlivých fázích, jak je obvykle prováděn. Jak jsem se již zmínil, není tento postup nikde standardizován a může být projekt od projektu odlišný. Uvádím zde jeho nejčastější podobu.

### 4.1.1 Odhad ve fázi Idea Formulation

Odhad v této fázi, tedy ve fázi počáteční, kdy je znám pouze jistý high level koncept požadovaného řešení, slouží primárně pro prioritizaci projektů na další období a pro kapacitní plánování.

Z toho, že je odhad prováděn na základě značně vágní specifikace požadavků, je jasné, že je to odhad spíše informativní a představuje pouze podklad pro utvoření představy o náročnosti projektu a zařazení do velikostní skupiny.

V této fázi se tvoří nejprve odhad na celý projekt a po případné prioritizaci projektu na následující rok se vytvoří i odhad na fázi Opportunity Note.

Odhad je vytvářen týmem globálních architektů. Po dohodě na něm mohou spolupracovat i architekti z oddělení, na které má projekt dopad. Odhad se provádí metodou Impact analýzy nad daným high level konceptem. Touto metodou se určí předpokládané dopady do ostatních systémů banky. Tyto se následně ohodnotí svou závažností jedním stupněm ze čtyř-stupňové škály. Pomocí tabulky s ohodnocením všech kombinací úrovní dopadu a všech aplikací se následně odhadne časová náročnost projektu. Tabulka byla sestavena na základě zkušeností a je průběžně upravována podle informací z dokončených projektů. Stejným způsobem se odhaduje hardwarová náročnost projektu.

Nové aplikace se ohodnocují obdobně. Určí se druh aplikace (jeden ze tří stupňů, - business critical, zvýšená pozornost, standart application) a opět podle příslušné tabulky pak určíme odhad.

Zdroje potřebné pro fázi Opportunity Note ( OpN) jsou standardně následující: IT Project Leader (IT PL), architekt, analytik a od každého dotčeného systému 2MD na konzultace.

#### 4.1.1.1 Odhad ve fázi Opportunity Note

Odhad se v této fázi vytváří pro fázi následující (Feasibility Study), je to tedy odhad na vypracování dokumentu IT Solution. Také se zpracovává přibližný odhad pro fázi projektovou, tedy na vývoj, testy a nasazení aplikace do provozu. Hlavním podkladem je dokument Expression of Needs (EoN) obsahující základní souhrn požadavků, které jsou na nový projekt kladeny. Dalším podkladem je výstup z této fáze (dokument s názvem Opportunity Note (OpN)), který obsahuje koncept řešení požadovaných funkčností. Jeho součástí jsou i odhadované náklady.

Odhad pro následující fázi projektu dodávají jednotlivé dotčené systémy (kompetenční centra). Používanou metodou bývá jistá forma zhodnocení dopadů a stanovení potřebné účasti na tvorbě IT Solution. Odhad nejčastěji tvoří pověřený analytik ve spolupráci s architektem.

Nástin odhadu pro projektovou fázi se pak určí tak, že se odhad pro feasibilitu vynásobí konstantou od 1 do 4. Ta se určí podle zhodnocení složitosti systému. Ve většině případů se odhad násobí číslem 3. Výsledná hodnota se prohlásí za odhad projektové fáze. Ten je používán pouze jako informativní s větší mírou nepřesnosti.

### 4.1.2 Odhad ve fázi Feasibility Study

Ve fázi Feasibility Study se odhad vytváří na celou projektovou fázi, tedy na vývoj, testy a nasazení aplikace do provozu. Hlavním podkladem jsou výstupy z této fáze, tj. primárně dokument s názvem IT Solution. Ten obsahuje detailnější návrh IT řešení projektu a je zároveň podkladem pro schválení projektové fáze.

Odhady dodávají jednotlivé dotčené systémy (kompetenční centra). Za tyto systémy tvoří odhady většinou architekt, případně analytik ve spolupráci s vývojáři a testery. Používaná metoda je většinou jistá forma individuálního či skupinového expertního úsudku, případně metoda založená na počítání use case.

Tvorba těchto odhadů není nijak metodicky standardizována.

## 4.2 SWOT Analýza současného stavu odhadování

Následující SWOT analýza se snaží nabídnout souhrn silných a slabých stránek výše popsaného způsobu odhadování a také přehled příležitostí a hrozeb, ze kterých jsem se snažil vycházet při tvorbě metodiky nové. Ta je podrobně popsána v kapitole 6.

### 4.2.1 Silné stránky

- Odhadování v počáteční fázi projektu je vzhledem k minimu informací poměrně dobře zpracované.
- Způsob odhadování je již silně zažitý.

### 4.2.2 Slabé stránky

- Neexistuje žádná standardizovaná metodika závazná pro všechny projekty.
- Z neexistence jednotné metodiky vyplývá neexistence standardizovaného způsobu měření velikosti projektů a měření efektivity vývoje.
- Prvotní ohodnocení projektu v počáteční fázi (Idea Formulation) je poměrně náročné a není jasné, zda je zvýšená náročnost vykoupena přesnějšími odhady.
- Absence vazby odhadů na historická data.

### 4.2.3 Příležitosti

- Změna metodiky odhadů v rámci přechodu na novou projektovou metodiku.
- Zavedení standardizovaného způsobu pro měření velikosti projektů a měření efektivity vývoje.
- Využití historických dat z projektů jako jednoho z podkladů pro tvorbu odhadů.

### 4.2.4 Hrozby

- Hrozba nepřesných odhadů vzhledem k neexistenci žádné závazné metodiky odhadů pro projekty.
- Změna projektové metodiky naruší zažité postupy odhadování.

# 5 Volba metod z hlediska vhodnosti pro Komerční banku

V následujících kapitolách bude uvedeno posouzení odhadovacích metod a ostatních atributů důležitých pro tvorbu ucelené metodiky odhadování projektů. Volbu metod provedu nejen s ohledem na využití ve vývoji Komerční banky, ale také s ohledem na jejich širší použitelnost.

## 5.1 Volba způsobu měření velikosti

Důležitou součástí metodiky odhadování je způsob měření velikosti projektů. Při volbě způsobu měření pro návrh metodiky jsem se rozhodoval mezi dvěma způsoby, a to mezi měřením pomocí řádek kódu a měřením pomocí funkčních bodů.

Hlavní výhodou řádků kódu je, že jsou jednoduše spočítatelné a velmi rozšířené. Naproti tomu jejich použití pro odhadování velikosti, zvláště v raných fázích projektu, může být poněkud neintuitivní a pro uživatele matoucí.

V Komerční bance se dosud žádná pevně stanovená metodika nepoužívala, odhady se tvořily přímo pomocí expertního úsudku a nebyla zde ani žádná základna nasbíraných dat s velikostmi projektů využívaná pro odhadování. Proto tedy nebyla volba žádným způsobem implikována. Zvolil jsem měření velikosti pomocí funkčních bodů. Důvodem pro tuto volbu bylo dle mého názoru intuitivnější používání FP v dřívějších fázích projektu. Domnívám se, že funkční pohled na velikost software více odpovídá realitě než LOC. Dalším důvodem pak byla existence metod pro odhadování FP v prvotních fázích projektu. Navíc před implementační fází můžeme funkční body z funkčních specifikací přímo spočítat a ne odhadovat, což u počítání řádků není možné.

Tento způsob měření velikosti budu aplikovat v průběhu celého projektu. Může být aplikován i na měření velikosti aplikačních celků a velikosti změn.

## 5.2 Volba odhadovacích metod

Vzhledem ke zvolenému způsobu měření velikosti projektu jsem volil mezi metodami pro odhadování funkčních bodů.

Pro jednotlivé fáze projektu jsou vhodné různé metody vzhledem k dostupnosti informací a požadavků na přesnost v těchto fázích. Protože se jednotlivé fáze projektu

odlišují rozpracovaností podkladů, specifikací požadavků a také požadavkem na přesnost výsledného odhadu, zvolil jsem pro každou fázi jiné metody.

V prvotní fázi projektu (Idea Formulation), kdy máme minimum informací a z toho důvodu je také požadavek na přesnost odhadu nejmenší, jsem zvolil jednu z metod odhadování pomocí zástupce, někdy se tomuto způsobu říká odhad pomocí fuzzy logiky.

V následující fázi je projekt více rozkrytý a máme již ucelenou představu, co se v jeho rámci bude řešit. Pro tuto fázi jsou vhodné metody popsané v kapitolách 3.2.1, 3.2.2, 3.2.3 a 3.2.4. Jsou to metody, které vyžadují již více informací a jejich přesnost je vyšší než u metody zvolené v předchozí fázi. Metody se liší v požadavcích na vstupní data pro odhadování, proto z důvodu možných odlišností v jednotlivých projektech ponechám volbu na tvůrci odhadu konkrétního projektu.

Ve fázi, která předchází samotné implementaci, bychom už mohli být schopni spočítat funkční body přímo, a proto je pro tuto fázi doporučena metoda přímého počítání funkčních bodů.

Převod odhadu velikosti ve funkčních bodech je nejlepší provádět pomocí historických dat, která se budou průběžně sbírat.

## 5.3 Historická data

Historická data z projektů jsou velmi důležitou součástí odhadování. V rámci metodiky jsem se rozhodl standardizovat sběr těchto dat z projektu a vytvořit jednotné úložiště těchto dat v databázi.

U každého projektu budeme uchovávat následující data:

- obecná data – obecná data o projektu (Název, IT PL …)
- klasifikace projektu – zařazení projektu do následujících kategorií. Vždy do jedné z každé skupiny:
  - nová aplikace, úprava stávající aplikace
  - implementační projekt, integrační projekt, infrastrukturní projekt, jiný
  - hardware, software, proces, jiný
- časový průběh projektu – časové vymezení jednotlivých fází projektu
- velikost projektu – velikost projektu uvedená ve funkčních bodech
- práce odvedená na projektu – práce odvedená na projektu v jednotlivých fázích rozdělená na jednotlivé role a podle příslušnosti zaměstnanců mezi interní a externí

- odhady na projekt – všechny odhady provedené na projekt v jednotlivých fázích

Tato data budou v rámci metodiky používána pro výpočet koeficientů pro převod velikosti na práci, předpokládané náklady a rozvrh projektu. Také budou průběžně používána pro kontrolu přesnosti odhadování a pro případné změny nastavení metod.

# 6 Návrh metodiky pro odhadování projektů

Přechod Komerční Banky (KB) na novou projektovou metodiku s sebou nese příležitost pro změnu metodiky odhadování projektů a s tím spojenou možnost zavedení standardu pro měření velikosti projektů a lepšího sledování efektivity práce. Zavedení nové metodiky pro odhadování není jen využití příležitosti, ale navíc jistý způsob reakce na nutnou změnu v zavedených, ovšem nestandardizovaných postupech odhadování.

Následující text obsahuje návrh metodiky pro odhadování projektů, který vychází z procesů KB, ale po menších úpravách by jistě mohl být použit i v jiných společnostech. Metodika je navíc podpořena nástrojem, který zajistí větší komfort pro odhadování, plánování a sledování průběhu projektů. Tento nástroj je podrobně popsán v kapitole 7 a je přiložen k této práci.

Metodika je určena výhradně pro odhadování nákladů za IT stranu řešení projektu, nezohledňuje a ani neodhaduje zdroje potřebné na straně klienta.

## 6.1 Základní principy

Základním principem a novým prvkem, který metodika zavádí, je měření velikosti projektů jednotným způsobem. To umožňuje srovnávání velikosti a měření efektivity vývoje. Je to také základ pro odhadovací metody používané pro odhadování v jednotlivých fázích projektu.

Zvoleno bylo měření velikosti pomocí funkčních bodů. Jeho podrobnější popis a rozbor vhodnosti naleznete v kapitole 2 a 5. Velikost projektu nebo jednotlivého úkolu je odhadnuta ve funkčních bodech a tato je následně převedena na náklady a potřebný čas na řešení pomocí koeficientů vypočtených z historických dat. Výpočet těchto koeficientů a jejich popis je uveden v kapitole 6.4.3.

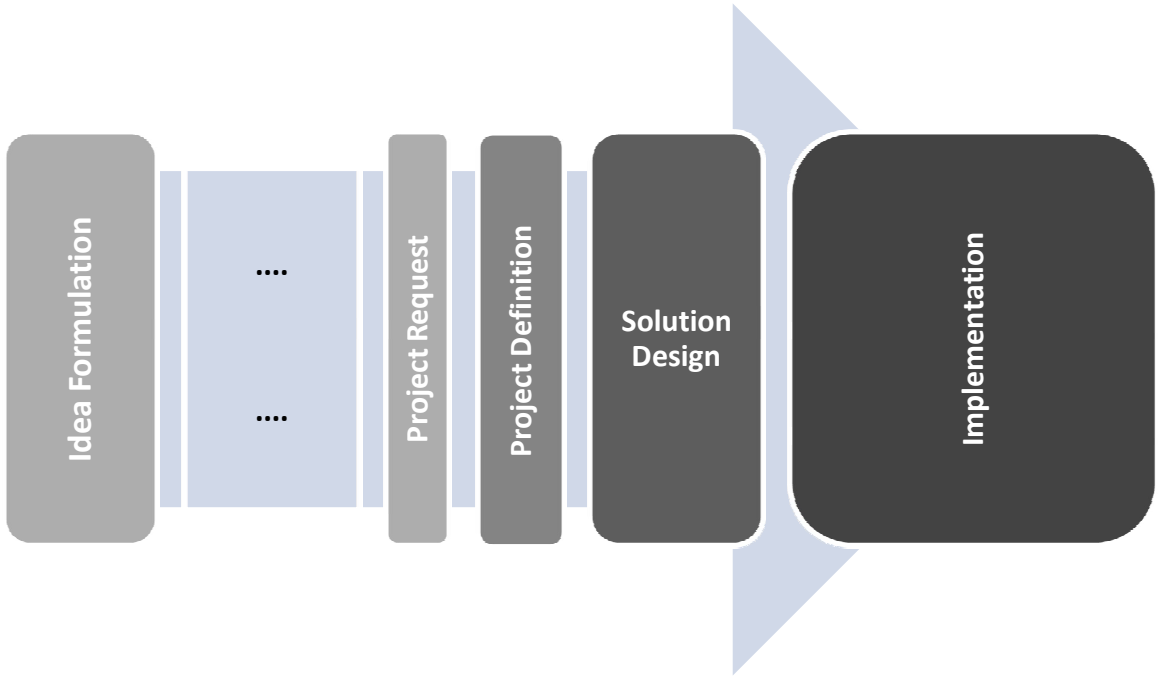
Dalším pilířem je databáze obsahující data z ukončených a běžících projektů. Tato databáze obsahuje cenná data, která se využívají jak pro odhadování velikosti projektu, tak pro převod této velikosti na náklady, časovou náročnost a rozdělení na jednotlivé role.

Další součástí databáze je katalog předpřipravených řešení, který obsahuje zpracovaná předpřipravená řešení. U každého je podrobný popis řešení a rozpis nákladů a času potřebného na implementaci. Vše je rozděleno po fázích a rolích.

Posledním pilířem metodiky je nástroj PRESTO (Project Effort Estimation Tool). Je to nástroj pro podporu odhadování projektů. Jeho součástí je výše popsaná databáze a jednoduché uživatelské rozhraní pro zadávání a aktualizaci dat o projektech v ní uložených a také rozhraní pro tvorbu odhadů na jednotlivé fáze. Použití tohoto nástroje v rámci metodiky je podrobněji popsáno v této kapitole a nástroj samotný je popsán v kapitole následující.

## 6.2 Projektová metodika

V současné době se v KB přechází na novou projektovou metodiku (KB a.s., 2008), která mino jiné mění počet fází projektu. Metodika odhadů popsaná v této kapitole tedy bude kopírovat fáze z nové metodiky. Následující obrázek znázorňuje jednotlivé fáze projektu.




**Obrázek 6.1 Fáze nové projektové metodiky KB. Diagram dle (KB a.s., 2008).**

Projektová metodika, tedy životní cyklus projektu, rozděluje projekt do několika fází. Nejprve se ve fázi Idea Formulation (IF) sepíše základní záměr a myšlenka projektu a je vytvořen první odhad nákladů potřebných na projekt. Tento koncept a náklady následně slouží jako podklady pro prioritizaci projektů na další rok.

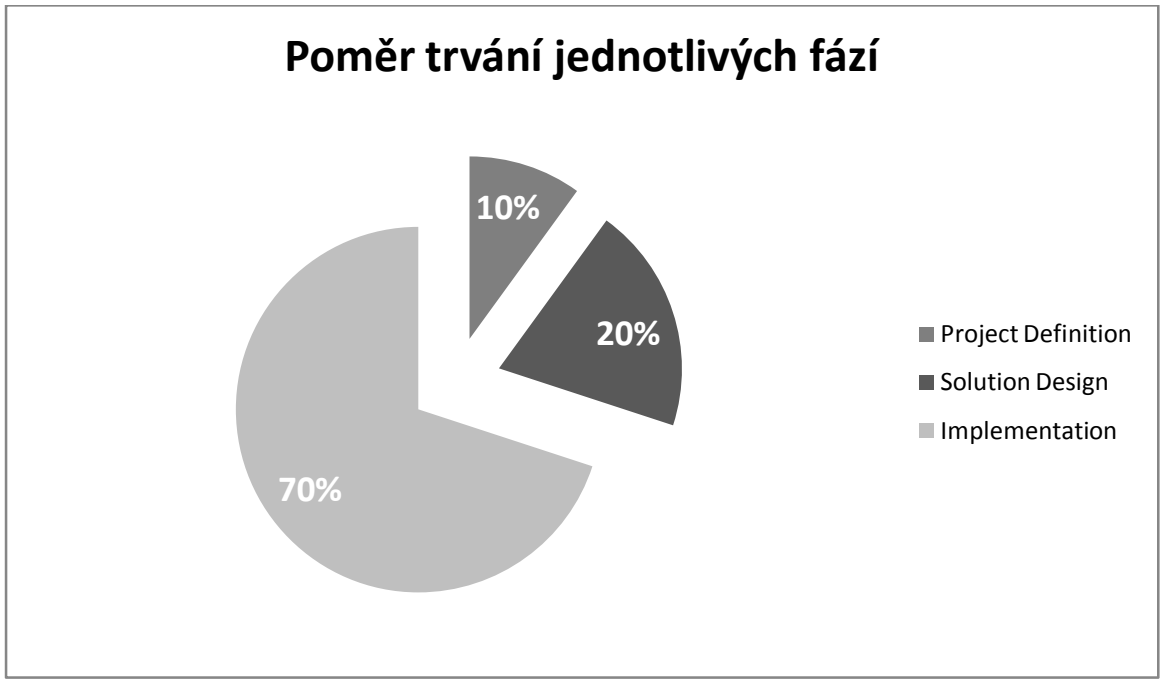
Pokud je projekt prioritizován, pak se v rámci fáze Project Request (PR) validují výstupy z fáze Idea Formulation a zohlední se případné změny. V případě dostupnosti zdrojů projekt přejde do fáze Project Definition (PD). Intenzivním zapojením menšího týmu je vytvořen dokument, který obsahuje podrobnější specifikaci požadavků a vyjasňuje většinu

problémů v projektu, aby mohl být odhad v této fázi co nejpřesnější. Odhad se stává totiž jedním z podkladů pro schválení přechodu projektu do další fáze.

Další fází je Solution Design (dále SD), ve které se připraví návrh IT řešení projektu a provede se další odhad na zbývající část životního cyklu projektu. Pokud tento odhad v součtu s již spotřebovanými náklady nevybočí z hranic tvořených intervalem +/- 50% okolo odhadu z fáze Project Definition, nemusí projekt znovu podstoupit schválení na Project Management Committee (PMC).

Poslední fází je Implementation. Zde po přípravě podrobných specifikací a dokumentace proběhne vývoj, otestování a nasazení produktu do produkčního prostředí.

V každé fázi je kromě souhrnného odhadu na všechny zbývající fáze projektu připraven také odhad na fázi bezprostředně následující a ten je pak použit pro naplánování této fáze.




**Obrázek 6.2 Předpokládaný poměr doby trvání jednotlivých fází projektu**

## 6.3 Metodika pro odhadování

V následujícím textu bude podrobně rozebrán návrh metodiky pro odhadování projektů pro KB. Nejprve jsou popsány jednotlivé fáze projektové metodiky, které korespondují s fázemi odhadování.


**Obrázek 6.3 Fáze projektu a přechody mezi nimi**

Na obrázku 6.3 jsou znázorněné jednotlivé fáze projektu a přechody mezi nimi. Tabulka 6.1 pak obsahuje stručné informace o aktivitách v jednotlivých fázích a podmínkách přechodu mezi nimi.

| Stádium | Hlavní aktivity | Přechod | Podmínky přechodu |
|---|---|---|---|
| Idea Formulation | Tvorba dokumentu Idea Formulation, který obsahuje základní popis myšlenky a záměr projektu. Rámcový odhad na celý projekt, který bude použit jako podklad pro prioritizaci projektů. | 1 | • záměr/myšlenka popsaná v dokumentu Idea Formulation.<br>• identifikován responsible banker a sponzor<br>• prioritizace projektu v rámci projektového plánu banky |
| Project Request | Validace a případné doplnění dokumentu Idea Formulation. Upravený rámcový odhad na celý projekt. | 2 | • stanovení IT PL, solution architekta a analytiků<br>• projekt ohodnocen z pohledu architektury<br>• schválení na IT ŘÍP |

| Stádium | Hlavní aktivity | Přechod | Podmínky přechodu |
|---|---|---|---|
| Project Definition | Tvorba dokumentu Project Definition, rozdělení projektů na menší úkoly a jejich seskupení do slotů a iterací. Identifikace možných náhradních řešení a jejich dopadů. Stanovení prvního přesnějšího odhadu projektu a alternativních řešení. | 3 | • validace dokumentu Project Definition na IT Architecture committee<br>• schválení na IT ŘÍP<br>• schválení na MPC/PMC |
| Solution Design | Tvorba dokumentu IT Solution. Rozdělení projektu na podrobný seznam úkolů. Popsání řešení jednotlivých úkolů a identifikace jejich nákladů. | 4 | • schválení na IT ŘÍP<br>• Pokud se odhad z fáze Project Definition pohybuje v rozmezí +/- 50% odhadu z fáze Project Definition, pak projekt může pokračovat bez schvalování na PMC/MPC. Jinak je potřeba schválení na této komisi. |
| Implementace | Tvorba detailních specifikací, vývoj, testování a nasazení produktu. | N/A | N/A |

**Tabulka 6-1 Proces vývoje softwaru s přechody**

| Přechod | Předpokládaná přesnost odhadu | Použití odhadu |
|---|---|---|
| 1 | Celý projekt: +/- 100% | Odhad je použit pro prioritizaci projektů v rámci kapacitního plánování. |

| Přechod | Předpokládaná přesnost odhadu | Použití odhadu |
|---|---|---|
| 2 | Celý projekt: +/- 100%<br>Další Fáze: +/- 20% | Upřesněný odhad na celý projekt slouží pro kapacitní plánování. Odhad na fázi Project Definition slouží k jejímu naplánování. |
| 3 | Celý projekt: +/- 50%<br>Další Fáze: +/- 10% | Odhad na celý projekt slouží jako podklad pro schválení. Odhad na další fázi slouží k jejímu naplánování. |
| 4 | Celý projekt: +/- 10%<br>Další Fáze: +/- 10% | Odhad slouží pro návrh rozpočtu projektové fáze. |

**Tabulka 6-2 Vazba přechodů mezi fázemi na odhady**

V tabulce 6-2 je přehled přechodů mezi jednotlivými projektovými fázemi a je zde nastíněna vazba na odhady, které jsou prováděny v jednotlivých fázích. Jsou zde uvedeny předpokládané přesnosti jednotlivých odhadů. Tyto přesnosti je nutné brát s rezervou, poněvadž jsou to čísla obvyklá pro úrovně detailu a rozpracovanosti řešení v jednotlivých fázích a nejsou zatím podložena historickou zkušeností s navrhovanou metodikou. Přesnější čísla budou moci být nabídnuta až po nějaké době provozu metodiky, případně po provedení studie na předchozích projektech. To s sebou, vzhledem ke změně metodiky projektového řízení, nese jisté obtíže s mapováním fází ze staré metodiky na nově používanou. Další těžkosti vznikají při počítání velikostí projektů ve funkčních bodech.

V následujících podkapitolách uvádím podrobnější přehled fází s popisem a přehledem vstupů a výstupů. U každé fáze navíc popisuji, co je již definováno, co o projektu známe a co můžeme při odhadování využít. Jsou zde také u každé fáze uvedeny všechny odhady, které jsou v jejím rámci vypracovávány. U odhadů je uveden stručný popis metody, podrobnější popisy metod jsou uvedeny dále.

### 6.3.1 Odhady ve fázi Idea Formulation

Ve fázi Idea Formulation se zformuluje základní záměr a myšlenka projektu. Odhadují se zde náklady na celý projekt a odhad následně slouží jako jeden z podkladů pro prioritizaci

projektů na následující rok. Vzhledem k nízké úrovni detailu zadání a poměrně velkému počtu neznámých je vytvořený odhad pouze informativní a má široký interval přesnosti.

**Metodika odhadu:**

*Odhad na celý projekt:* Metoda založená na expertním odhadu a využívající historická data. Způsob odhadování je podrobně popsán v kapitole 6.5.1.

**Vstupy:**

- dokument Idea Formulation

**Je definováno:**

- základní záměr a myšlenka projektu

**Výstupy:**

- První rámcový odhad nákladů na celý projekt. Odhad počtu MD a nákladů na jednotlivé fáze s rozdělením na externí, interní a na analýzu, architekturu, vývoj, testy a management.
- Rámcový odhad prvního roku poimplementačních nákladů na podporu.

Vzhledem k počáteční fázi specifikace požadavků a obecnému rázu dokumentu Idea Formulation, je odhad v této fázi pouze informativní a má široký interval přesnosti. Odhad slouží primárně pro kapacitní plánování a není určen pro plánování projektu.

#### 6.3.1.1 Odhad na celý projekt

Odhad na celý projekt se provede pomocí metody pro odhadování počtu funkčních bodů popsané v kapitole 6.5.1.

Uvedená metoda je podporovaná nástrojem PRESTO. Pod volbou odhadování ve fázi Idea Formulation se nachází formulář na zadání potřebných údajů a po jeho vyplnění nástroj kalkuluje odhad.

Nástroj jako výsledek odhadování nabídne rozpad zdrojů na jednotlivé role a typ zdrojů v textové i grafické podobě. Výsledný odhad a rozpad zdrojů je vzhledem k nízké úrovni rozpracovanosti zadání pouze rámcový a neměl by být podkladem pro plánování projektu. Odhad je určen pouze pro potřeby prioritizace projektů a kapacitní plánování.

### 6.3.2 Odhady ve fázi Project Request

Celá fáze slouží primárně k validaci změn mezi IF a PR. Dopad a změny se ohodnocují podle stejných pravidel jako ve fázi IF.

**Metodika odhadu:**

*Odhad na celý projekt:* Metoda založená na expertním odhadu a využívající historická data shodná s metodou odhadu provedeného ve fázi Idea Formulation.

*Odhad na fázi Project Definition:* Odhad je určen z odhadu na celý projekt a validován pomocí expertního odhadu.

**Vstupy:**

- aktualizovaný odhad Idea Formulation

**Je definováno:**

- základní záměr a myšlenka projektu

**Výstupy:**

- aktualizovaný odhad na celý projekt a odhad na fázi Project Definition

V odhadech je uveden celkový počet MD a celkové náklady na řešení a také rozpad MD na jednotlivé role a fáze. Je zde rovněž rozdělení na externí a interní MD a náklady na případný hardware.

#### 6.3.2.1 Odhad na celý projekt

Pokud se od fáze Idea Formulation nějak změnilo zadání projektu (tedy pokud se nějak změnil dokument Idea Formulation), pak je nutné odhadnout celý projekt znovu, a to stejným způsobem jako v předchozí fázi. Postup je podrobně popsán v kapitole 6.5.1.

Jestliže se dokument Idea Formulation nezměnil, není potřeba projekt znovu odhadovat a odhad z fáze Idea Formulation je stále platný.

Tento odhad je použit zejména pro kapacitní plánování. Co se přesnosti týče, platí pro něj to samé co pro odhad z fáze Idea Formulation.

#### 6.3.2.2 Odhad na fázi Project Definition

Z odhadu celého projektu provedeného v této fázi, nebo případně převzatého z fáze Idea Formulation, získáme velikost projektu odhadnutou ve funkčních bodech. Pomocí koeficientů získaných z historických dat, které jsou definovány a popsány v kapitole 6.4.3, vypočteme potřebné náklady na fázi Project Definition , rozpočteme odhad na jednotlivé role a rozdělíme je na externí a interní.

Tento proces vytvoření odhadu je zautomatizován v nástroji PRESTO. Tento nástroj nám v případě, že je v něm uložen aktuální odhad na celý projekt, na vyžádání vydá textovou i grafickou podobu odhadu na fázi Project Definition včetně rozpadu na jednotlivé role a typ zdrojů.

Výsledný odhad a rozpad zdrojů může být podkladem pro naplánování fáze nebo případně ještě validován nominovaným IT PL a architektem. Případné větší odlišnosti budou muset být zdůvodněny.

Standardně se na této fázi podílejí tyto IT role:

- projektový manažer
- globální architekt
- architekt kompetenčního centra
- hlavní analytik kompetenčního centra
- architekt podpory (oddělení 5320)
- architekt infrastruktury (oddělení 5340)

### 6.3.3 Odhady na fázi Project Definition

Tato fáze slouží k tomu, aby se pomocí intenzivní spolupráce podařilo zafixovat rozsah projektu, stanovit trvání, náklady, přínosy a určit pracnost projektu. Dále je nutné rozdělit projekt na dílčí úkoly a ty následně seskupit do slotů a iterací. Pokud je to možné, je velmi důležité nalézt, analyzovat a vyhodnotit alternativní řešení a případně doporučit jedno z nich.

Na základě těchto výstupů je následně proveden odhad, který slouží jako podklad pro schvalování projektu. Je snaha odhad, za daných podmínek a míry rozpracovanosti zadání, utvořit s co největší přesností. Z celkového odhadu je pak utvořen i odhad na následující fázi, tedy Solution Design.

**Metodika odhadu:**

*Odhad na celý projekt:* Odhad je tvořen metodou odhadu funkčních bodů. Metody jsou blíže popsané v kapitole 6.5.3.

*Odhad na fázi Solution Design:* Odhad je tvořen expertním úsudkem a konfrontován s odhadem na celý projekt.

**Vstupy:**

- dokument Project Definition
- odhad na projekt z minulé fáze

**Je definováno:**

- procesy, jejich četnost, doba trvání a návaznosti
- přibližný počet uživatelů
- množství dat (smluv, obchodních případů …)
- jsou známy návaznosti do jiných business procesů
- požadavky na dostupnost
- požadavky na podporu
- požadavky na historii dat
- požadavky na archivaci
- existuje rozdělení požadavků na:
    - nezbytné jádro
    - nutné z hlediska smysluplného využití
    - volitelné, realizovatelné v případě dostatečných zdrojů
    - známá časová a finanční omezení

**Výstupy:**

- přesný odhad na následující fázi, tedy fázi Solution Design a také hrubý odhad na celý projekt.

Odhad na fázi Solution Design je s přesností +/- 20%. Odhad na celý projekt pak s přesností blížící se k +/- 50%.

#### 6.3.3.1 Odhad na celý projekt

Odhad na celý projekt se provede pomocí metod pro odhadování počtu funkčních bodů uvedených v kapitole 6.5.3. Odhad se může provádět na celý scope projektu, ale pokud existuje rozdělení scope na úkoly, je možné a vhodné je odhadovat zvlášť. U každého takového úkolu je žádoucí provést odhad jiným způsobem, například je možné jeden úkol spočítat přímo a další odhadnout. Velmi výhodné může být použití seznamu předpřipravených řešení, která se mohou přiřadit úkolům a tím je odhadnout. Toto rozdělování na úkoly také nástroj PRESTO plně podporuje.

Existuje možnost výběru mezi více metodami odhadu funkčních bodů. Pracovníci pověřeni provedením odhadu jsou povinni se s popsanými metodami seznámit a podle dostupných dat a vlastností metod zvolit tu nejvhodnější.

Metodikou je doporučeno identifikovat co nejvíce úkolů, které lze odhadnout pomocí analogie, tedy jim přiřadit některé z předpřipravených řešení v databázi. Mezi ostatními úkoly pak definovat ty, u kterých lze spočítat funkční body přímo, a zbylé úkoly odhadnou metodou zvolenou na základě dostupnosti dat.

Všechny metody uvedené v kapitole 6.5.3 jsou podporovány nástrojem PRESTO. Více o způsobu odhadování pomocí tohoto nástroje je napsáno v nápovědě k nástroji a na přiloženém CD.

Nástroj, jako výsledek odhadování, nabídne rozpad zdrojů na jednotlivé role a typy v textové i grafické podobě. Výsledný odhad a rozpad zdrojů může být podkladem pro naplánování fáze nebo ještě validován nominovaným IT PL a architektem. Případné větší odlišnosti budou muset být zdůvodněny.

#### 6.3.3.2 Odhad na fázi Solution Design

Z odhadu celého projektu provedeného v této fázi získáme velikost projektu odhadnutou ve funkčních bodech. Pomocí koeficientů získaných z historických dat, která jsou definována a popsána v kapitole 6.5.3, vypočteme potřebné náklady na fázi Solution Design a také rozpočteme odhad na jednotlivé role a rozdělíme je na externí a interní zdroje. Tento proces vytvoření odhadu je zautomatizován v nástroji PRESTO.

Výsledný odhad a rozpad zdrojů může být podkladem pro naplánování fáze nebo ještě validován nominovaným IT PL a architektem. Případné větší odlišnosti budou muset být zdůvodněny.

### 6.3.4 Odhady ve fázi Solution Design

Z odhadu celého projektu provedeného v této fázi nebo převzatého z fáze Idea Formulation získáme velikost projektu odhadovanou ve funkčních bodech. Pomocí historických dat tento údaj přepočteme na počet dní práce potřebných pro jednotlivé role.

Výsledek pak bude validován nominovaným IT PL, architektem a analytikem. Případné větší odlišnosti budou muset být zdůvodněny.

**Metodika odhadu:**

*Odhad na celý projekt:* Zpřesnění odhadu z předešlé fáze pomocí metody počítání funkčních bodů 6.5.2. Případně odhad počtu funkčních bodů jednou z metod uvedených v kapitole 6.5.3.

*Odhad na fázi Solution Design:* Odhad je tvořen expertním úsudkem a konfrontován s odhadem na celý projekt.

**Vstupy:**

- dokument Solution Design
- odhad počtu funkčních bodů z fáze Project Definition

**Je definováno:**

- rozpracované požadavky
- rozpracované procesy
- rozpracovaná architektura řešení
- vše na úrovni dokumentu IT Solution
- rozdělení scope projektu na jednotlivé úkoly

**Výstupy:**

- odhad na fázi Implementation

#### 6.3.4.1 Odhad na fázi Implementation

Odhad na fázi Implementation se provede metodou přímého spočítání funkčních bodů aplikace podle postupu uvedeného v kapitole 6.5.2. Přímé počítání se provede, pouze pokud je pro něj dostatek podkladů. Pokud tomu tak není, je velikost ve funkčních bodech odhadnuta pomocí metod uvedených v kapitole 6.5.3. Odhad se může provádět na celý scope

projektu, ale vzhledem k pokročilé fázi specifikace je mnohem lepší projekt rozdělit na více úkolů a tyto jednotlivé úkoly odhadovat zvlášť. U každého takového úkolu je možné provést odhad jiným způsobem, například je možné jeden úkol spočítat přímo a další odhadnout. Velmi výhodné může být použití seznamu předpřipravených řešení, která se mohou přiřadit úkolům a tím je odhadnout. Toto rozdělování na úkoly také nástroj PRESTO plně podporuje.

Je zde možnost výběru mezi více metodami odhadu funkčních bodů. Pracovníci pověřeni provedením odhadu jsou povinni se s popsanými metodami seznámit a podle dostupných dat a vlastností metod zvolit tu nejvhodnější.

Metodikou je doporučeno identifikovat co nejvíce úkolů, u kterých lze velikost spočítat přímo. Zbylé úkoly odhadnout metodou zvolenou na základě dostupnosti dat, které vyžaduje.

Všechny metody uvedené v kapitole 6.5.3 jsou podporované nástrojem PRESTO. Nástroj, jako výsledek odhadování, nabídne rozpad zdrojů na jednotlivé role a typ zdrojů v textové i grafické podobě. Výsledný odhad a rozpad zdrojů může být podkladem pro naplánování fáze nebo ještě validován nominovaným IT PL a architektem. Případné větší odlišnosti budou muset být zdůvodněny.

### 6.3.5 Fáze Implementation

Po skončení projektové fáze se pomocí metody počítání funkčních bodů podrobně popsané v kapitole 6.5.2 spočte skutečná velikost projektu. Tato hodnota spolu s hodnotou vyčerpaných nákladů je uložena do databáze. Následně se provede vyhodnocení přesnosti odhadů vytvořených během jednotlivých fází. U případných větších než standardních odlišností se provede vyhodnocení příčiny a navržení možné korekce odhadování.

## 6.4 Data pro podporu odhadování

Součástí používaných metod pro odhadování jsou historická data a parametry z nich kalkulované. Tato data jsou uchovávána ve speciální databázi. Její popis se nachází v následujících kapitolách.

### 6.4.1 Historická data

Historická data obsahují data o již uzavřených i běžících projektech. O projektech si uchováváme přehledy čerpaných nákladů a přehled dob trvání jednotlivých fází. Čerpané MD bychom měli mít rozděleny po jednotlivých rolích a na externí a interní. Uchováváme také

všechny odhady, které jsme v jednotlivých fázích provedli pro budoucí porovnání se skutečností a pro možnost ohodnocení přesnosti jednotlivých odhadů.

Struktura historických dat je následující:

- obecné informace o projektu
    - jméno projektového manažera (IT PL)
    - jméno projektu
    - stručný popis projektu
    - typ projektu (HW, SW, proces)
    - kompetenční centrum nebo aplikace, do které projekt zasahuje
    - druh projektu: nová aplikace, úprava stávající aplikace, integrační projekt, implementační projekt, infrastrukturní projekt, proces
- odhadované časové vymezení jednotlivých fází a náklady
    - odhadované doby trvání fází z odhadů v jednotlivých fázích projektu
    - odhadované náklady a počty MD na řešení
- skutečné časové vymezení jednotlivých fází
    - data schválení jednotlivých fází
    - datum nasazení projektu
    - datum ukončení projektu
    - doba trvání jednotlivých fází
- skutečné náklady a počty MD na jednotlivé fáze
    - náklady a počty MD na jednotlivé fáze rozdělené podle pracovníků na interní a externí a dále podle zaměření na analýzu, vývoj, testy, podporu a management
    - náklady na HW
- parametry z odhadovacích metod
    - parametry nastavení metod při odhadování v jednotlivých fázích
    - parametry pro tvorbu odhadu ve fázi IF

**Generované statistiky z projektových dat:**

- poměry rozdělení nákladů mezi interní, externí zdroje a mezi analýzu, vývoj, testy, management, podporu …
- koeficienty určené pro odhadovací metody

### 6.4.2 Ostatní data

Mimo historická data databáze obsahuje také obecná data potřebná pro převod odhadu v MD na náklady. Zahrnuta jsou rovněž data o předpřipravených řešeních, která lze přiřazovat jednotlivým úkolům v projektu, a tím odhadovat jejich velikost.

#### 6.4.2.1 Obecná data

Obecná data obsahují údaje potřebné pro výpočet nákladů na projekty a ostatní data využitelná například pro kapacitní plánování. Přehled dat je uveden níže.

- průměrná cena za MD (analýza, vývoj, testy, IT PL)
    - rozděleno podle externí / interní
    - rozděleno na role
- cena za HW přes všechny kategorie dostupnosti a platformu
    - databázový server
    - aplikační server
    - web server
    - uživatelské PC
    - ostatní HW
- poměry rozložení pracností mezi jednotlivé fáze pro všechny druhy projektů – budou generovány z historických dat

#### 6.4.2.2 Předpřipravená řešení

V řadě aplikací existují množiny činností, které jsou do značné míry unifikované, nebo unifikovatelné. Jde o funkčnosti spojené s user managementem, zálohováním, archivací, auditem, využitím existujících procesů (služeb) atd.

U každé z těchto funkčností lze stanovit standardní cenu a dobu realizace pro další použití. V případě, že zadavatel zná vlastnosti těchto funkčností, není nutné, aby je detailně specifikoval. Musí si být ovšem vědom toho, že by měl tyto služby použít tak, jak jsou.

Speciálním případem opakovaných řešení jsou otázky HW, kdy je třeba mít jednoznačné informace o cenách HW vztažených k objemu dat a charakteru aplikace, včetně nákladů na jeho zprovoznění.

Pro každé takové opakované řešení jsou v databázi zaznamenány následující informace:

- podrobný popis řešení

- typ řešení (HW, SW, proces)
- funkční a nefunkční vlastnosti řešení
- náklady a MD na Project Definition
- náklady a MD na Solution Design
- náklady a MD na Implementation
- provozní náklady
- velikost řešení ve funkčních bodech

### 6.4.3 Generované koeficienty a přepočítání velikosti na náklady

Na základě předpokladu, že vývoj projektů s podobnými vlastnostmi bude probíhat s podobnou produktivitou a s podobnou skladbou nákladů, můžeme z historických dat projektů vypočítat průměrnou cenu jednoho funkčního bodu v MD nebo přímo v nákladech.

Jelikož však existuje řada různých typů projektů a tyto se můžou v produktivitě výrazněji lišit, je výhodnější nepočítat koeficienty pro přepočet funkčních bodů ze všech projektů, ale pouze z projektů s podobnými vlastnostmi. Za tímto účelem bylo zvoleno několik parametrů projektu, které tvoří jistou diskrétní míru podobnosti. Definice této míry je uvedena v následující podkapitole.

Dále je pak uveden přehled a popis koeficientů, které se používají pro převod funkčních bodů na odhady jednotlivých fází a rozpis MD a nákladů na jednotlivé role.

#### 6.4.3.1 Definice podobnosti projektů

Každý projekt má několik atributů, pomocí nichž můžeme definovat podobnost projektů. Nejprve však několik obecných definic:

- *f* označuje fázi projektu a platí $f \in \{PD, SD, I\}$, kde
  - *PD* je Project Definition
  - *SD* je Solution Design a
  - *I* je Implementation
- *z* označuje typ zdroje a platí $z \in \{EXT, INT\}$, kde
  - *EXT* je externí zdroj
  - *INT* je interní zdroj
- *r* označuje roli v projektu a platí $r \in \{PM, AN, ARCH, TEST, DEV, 20, 40\}$, kde

    - *PM* je projekt management
    - *AN* je analýza
    - *ARCH* je architektura
    - *TEST* jsou testy
    - *DEV* je vývoj
    - *20* je podpora (oddělení 5320)
    - *40* je infrastruktura (oddělení 5340)
- *N* je počet ukončených projektů v databázi. Tyto projekty pak mají pořadová čísla *1..N*
- $NK_v$ – náklady na projekt s pořadovým číslem *v* (náklady na celý projekt)
- $NK_{v,f}$ – náklady na projekt s pořadovým číslem *v* (náklady na fázi *f*)
- $NK_{v,z}$ – náklady na projekt s pořadovým číslem *v* (náklady na typ zdroje *z*)
- $NK_{v,r}$ – náklady na projekt s pořadovým číslem *v* (náklady na projektovou roli *r*)
- $NK_{v,f,z}$ – náklady na projekt s pořadovým číslem *v* (náklady na fázi *f* a typ zdroje *z*)
- $NK_{v,f,r}$ – náklady na projekt s pořadovým číslem *v* (náklady na fázi *f* a projektovou roli r)
- $NK_{v,z,r}$ – náklady na projekt s pořadovým číslem *v* (náklady na projektovou roli *r* a typ zdroje *z*)
- $NK_{v,f,z,r}$ – náklady na projekt s pořadovým číslem *v* (náklady na fázi *f,* typ zdroje *z* a projektovou roli *r*)
- $MD_v$ – počet MD na projekt s pořadovým číslem *v* (náklady na celý projekt)
- $MD_{v,f}$ – počet MD na projekt s pořadovým číslem *v* (náklady na fázi *f*)
- $MD_{v,z}$ – počet MD na projekt s pořadovým číslem *v* (náklady na typ zdroje *z*)
- $MD_{v,r}$ – počet MD na projekt s pořadovým číslem *v* (náklady na projektovou roli *r*)
- $MD_{v,f,z}$ – počet MD na projekt s pořadovým číslem *v* (náklady na fázi *f* a typ zdroje *z*)
- $MD_{v,f,r}$ – počet MD na projekt s pořadovým číslem *v* (náklady na fázi *f* a projektovou roli *r*)
- $MD_{v,z,r}$ – počet MD na projekt s pořadovým číslem *v* (náklady na projektovou roli *r* a typ zdroje *z*)
- $MD_{v,f,z,r}$ – počet MD na projekt s pořadovým číslem *v* (náklady na fázi *f,* typ zdroje *z* a projektovou roli *r*)

Atributy pro definici podobnosti jsou následující:

- Vlastnost *CC* značí kompetenční centrum, ve kterém je projekt realizován (nabývá hodnot: DCS, CMS, BDC, SSC, BI, TSS, ITT…).
- Vlastnost *DP1* značí druh projektu I (nabývá hodnot: implementační projekt, infrastrukturní projekt, procesní projekt, integrační projekt).
- Vlastnost *DP2* značí druh projektu II (nabývá hodnot: úprava stávající funkčnosti, zavedení nové funkčnosti).
- Vlastnost *VP* značí velikost projektu ve funkčních bodech (velikost udaná v počtu funkčních bodů).

Dále platí následující definice:

- Pro projekty *u* a *v* označíme jako $pCC_{u,v}$ podobnost hodnot $CC_u$ a $CC_v$ a platí následující:

  Když $CC_u = CC_v$ pak $pCC_{u,v} = 1$ jinak $pCC_{u,v} = 0$

- Pro projekty *u* a v označíme jako $pDP1_{u,v}$ podobnost hodnot $DP1_u$ a $DP1_v$ a platí následující:

  Když $DP1_u = DP1_v$ pak $p\ DP1_{u,v} = 1$ jinak $p\ DP1_{u,v} = 0$

- Pro projekty *u* a *v* označíme jako $pDP2_{u,v}$ podobnost hodnot $DP2_u$ a $DP2_v$ a platí následující:

  Když $DP2_u = DP2_v$ pak $pDP2_{u,v} = 1$ jinak $pDP2_{u,v} = 0$

- Pro projekty *u* a *v* označíme jako $pVP_{u,v}$ podobnost hodnot $VP_u$ a $VP_v$ a platí následující:

  Když *VPu* z intervalu $<VP_v - c;\ VP_v + c>$, pak $pVP_{u,v} = 1$ jinak $pVP_{u,v} = 0$, kde

  $c = (maxVP - minVP)/8$

  *maxVP* je maximální velikost z projektů v databázi

  *minVP* je minimální velikost z projektů v databázi

Pomocí těchto definic můžeme následně definovat podobnost projektů. Označme $D_{u,v}$ jako podobnost projektů *u* a *v*. Pro $D_{u,v}$ platí následující vztah:

$$D_{u,v} = pCC_{u,v} * pDP1_{u,v} * pDP2_{u,v} * pVP_{u,v}$$

Takto definovanou podobnost projektů používám ve výpočtu koeficientů uvedených v následujících kapitolách. Důvodem pro použití podobnosti je snaha o zpřesnění výpočtu koeficientů zohledněním specifických vlastností projektů, které se mohou projevit na produktivitě v rámci daného projektu.

Vzorce na výpočet koeficientů pro přepočet funkčních bodů, uvedené dále v textu, používají následující pojmy:

- *P* je index projektu, pro který vypočítáváme koeficienty
- $FP_v$ je počet funkčních bodů projektu s indexem *v*

#### 6.4.3.2 Koeficienty pro výpočet celkových nákladů a MD

*Koeficient pro výpočet celkových nákladů:*

$$kNK = \frac{\sum_{k=1}^{N} \frac{NK_k}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet celkových MD:*

$$kMD = \frac{\sum_{k=1}^{N} \frac{MD_k}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

#### 6.4.3.3 Koeficienty pro výpočet nákladků a MD na jednotlivé fáze

*Koeficient pro výpočet nákladů na Project Definition:*

$$kNK_{PD} = \frac{\sum_{k=1}^{N} \frac{NK_{k,PD}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD na Project Definition:*

$$kMD_{PD} = \frac{\sum_{k=1}^{N} \frac{MD_{k,PD}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet nákladů na Solution Design:*

$$kNK_{SD} = \frac{\sum_{k=1}^{N} \frac{NK_{k,SD}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD na Solution Design:*

$$kMD_{SD} = \frac{\sum_{k=1}^{N} \frac{MD_{k,SD}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet nákladů na Implementation:*

$$kNK_{I} = \frac{\sum_{k=1}^{N} \frac{NK_{k,I}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD na Implementation:*

$$kMD_{I} = \frac{\sum_{k=1}^{N} \frac{MD_{k,I}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

#### *6.4.3.4* Koeficienty pro výpočet nákladů a MD na jednotlivé role

*Koeficient pro výpočet MD za projektový management:*

$$kMD_{PM} = \frac{\sum_{k=1}^{N} \frac{MD_{k,PM}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za analýzu:*

$$kMD_{AN} = \frac{\sum_{k=1}^{N} \frac{MD_{k,AN}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za architekturu:*

$$kMD_{ARCH} = \frac{\sum_{k=1}^{N} \frac{MD_{k,ARCH}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za vývoj:*

$$kMD_{DEV} = \frac{\sum_{k=1}^{N} \frac{MD_{k,DEV}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za testy:*

$$kMD_{TEST} = \frac{\sum_{k=1}^{N} \frac{MD_{k,TEST}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za podporu (5320):*

$$kMD_{20} = \frac{\sum_{k=1}^{N} \frac{MD_{k,20}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za infrastrukturu (5340):*

$$kMD_{40} = \frac{\sum_{k=1}^{N} \frac{MD_{k,40}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

#### 6.4.3.5 Koeficienty pro výpočet nákladů a MD na interní a externí zdroje

*Koeficient pro výpočet nákladů na interní zdroje:*

$$kNK_{INT} = \frac{\sum_{k=1}^{N} \frac{NK_{k,INT}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet nákladů na externí zdroje:*

$$kNK_{EXT} = \frac{\sum_{k=1}^{N} \frac{NK_{k,EXT}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za interní zdroje*

$$kMD_{INT} = \frac{\sum_{k=1}^{N} \frac{MD_{k,INT}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

*Koeficient pro výpočet MD za externí zdroje:*

$$kMD_{EXT} = \frac{\sum_{k=1}^{N} \frac{MD_{k,EXT}}{\mathrm{FP}_k} * D_{k,p}}{\sum_{k=1}^{N} D_{k,p}}$$

#### 6.4.3.6 Přepočítání funkčních bodů na náklady a pracnost

S pomocí koeficientů uvedených v předchozí kapitole můžeme z počtu funkčních bodů odhadnout náklady a počet MD. Můžeme vypočítat také náklady a MD jen za určitou fázi, projektovou roli, typ zdroje nebo jejich kombinaci.

Definujeme:

- $oNK_{p,[f],[z],[r]}$ – náklady vypočtené z funkčních bodů na projekt s pořadovým číslem *p* (náklady mohou být omezeny na fázi *f,* typ zdroje *z,* projektovou roli *r* a nebo na jejich kombinaci)
- $oMD_{p,[f],[z],[r]}$ – MD vypočtené z funkčních bodů na projekt s pořadovým číslem *p* (MD mohou být omezeny na fázi *f,* typ zdroje *z* a projektovou roli *r* a nebo na jejich kombinaci)

MD nebo náklady na projekt s pořadovým číslem *p* pak vypočteme podle vzorce

$$oNK_{p,[f],[z],[r]} = FP_p * kNK_{[f],[z],[r]}$$

pro výpočet nákladů a

$$oMD_{p,[f],[z],[r]} = FP_p * kMD_{[f],[z],[r]}$$

pro výpočet MD.

Ve vzorcích lze použít všechny možné kombinace projektových fází, typů zdrojů a projektových rolí. Je tedy možné použít právě tu kombinaci a vypočítat to, co potřebujeme.

### 6.4.4 Počáteční naplnění databáze

V rámci KB existují jisté informace o průběhu minulých projektů, uložené ve formě reportů z jednotlivých projektů. Jsou zde uloženy informace o počtu čerpaných MD, nákladech a době trvání jednotlivých fází a celého projektu. V jejich použití pro navrhovanou metodiku nám však brání několik skutečností. Prvním problémem je to, že projekty byly prováděny ještě dle staré metodiky, a tedy fáze měly jinou dobu trvání a byl i jiný počet fází. Proto jsou data o fázích nepoužitelná. Dalším problémem, a to ještě závažnějším je, že projekty neobsahují ohodnocení velikosti pomocí funkčních bodů. Nemáme tedy data o produktivitě a převodu velikosti na práci.

Poněvadž nemáme historická data, na jejichž základě bychom odhadovali, je nutné použít data obecná. Pro převod funkčních bodů na MD použijeme převodní tabulku z (McConnell, 2006). Databáze zůstane na začátku prázdná a budou se z počátku používat obecné koeficienty pro převod velikosti na práci.

Historická data se budou průběžně sbírat, a jakmile jich bude dostatek, vypočítáme koeficienty přímo z databáze. Odhady budou na začátku méně přesné a postupem času se budou zpřesňovat.

Jediná historická data, která jsou použitelná, jsou celková suma nákladů a MD z projektů. Tato data lze následně využít při odhadování ve fázi IF, za předpokladu že budeme v této fázi odhadovat přímo práci a náklady, a ne funkční body.

## 6.5 Detailní popis metod

V této kapitole jsou uvedeny podrobné popisy metod použité v rámci metodiky. Některé pojmy a postupy navazují na popisy metod v kapitole 2, proto je vhodné přečíst si nejprve onu kapitolu a pak pokračovat zde.

Všechny popsané metody jsou podporovány nástrojem PRESTO.

### 6.5.1 Odhad ve fázi Idea Formulation

V této fázi jde hlavně o základní stanovení rozsahu projektu, který je jen informativní s malou přesností. To je způsobeno velmi omezeným rozpracováním požadavků a spíše vágními formulacemi v dokumentu Idea Formulation.

Odhad provede tým globálních architektů s případným přispěním architektů z dotčeného kompetenčního centra.

Projekt se před odhadováním zanese do systému a budou mu nastaveny základní vlastnosti, které ovlivňují odhadování. Tyto vlastnosti jsou následující:

- kompetenční centrum, na které má projekt hlavní dopad
- atribut, který značí, zda jde o novou nebo stávající aplikaci

Následně se z pětibodové škály přiřadí stupeň odpovídající náročnosti projektu. Tato škála je následující:

1) velmi malý
2) malý
3) střední
4) velký
5) velmi velký

Z databáze dokončených projektů se vyberou projekty se shodně nastavenými hodnotami výše popsaných atributů. Pokud počet takto vybraných projektů není alespoň deset, pak zvolíme všechny projekty z databáze. U vybraných projektů nás zajímá velikost projektu, tedy hodnota funkčních bodů. Z této množiny hodnot vypočítáme odhadovanou velikost projektu podle zvolené velikosti tak, že platí následující:

1) Velmi malý = 10. percentil z vybraných hodnot
2) Malý = 25. percentil z vybraných hodnot

3) Střední = 50. percentil z vybraných hodnot
4) Velký = 75. percentil z vybraných hodnot
5) Velmi velký = 90. percentil z vybraných hodnot

Pro velmi malý projekt tedy platí, že 10% projektů má menší velikost a podobně pak pro ostatní. Výsledná čísla následně zaokrouhlíme na stovky, abychom předešli mylné interpretaci přesnosti odhadu.

Odhaduje se rovnou potřebná práce a náklady na projekt. Celková čísla se poté pomocí koeficientů převedou na jednotlivé role a fáze. V případě existence dat, ze kterých se určí očekávaná přesnost této metody, se může u odhadu uvádět také interval vyjadřující tuto přesnost.

### 6.5.2 Počítání funkčních bodů

Co jsou to funkční body a proč je vůbec používáme, je popsáno v prvních kapitolách této práce. Nyní uvedu popis toho, jak určit počet funkčních bodů projektu či aplikace.

Nejprve ve stručnosti naznačím postup:

- Identifikujte a spočtěte ILF, EIF, EI, EO, EQ. Pro každou ILF a EIF identifikujte počet RET a počet DET. Pro každou EI, EO a EQ identifikujte počet FTR (File Type Referenced) a DET (Data Element Type).
- S použitím matice složitosti spočtěte počet jednoduchých, průměrných a složitých položek EI, EO, EQ, ILF, EIF.
- Spočtěte počet neupravených funkčních bodů.
- Určete hodnoty 14 charakteristik systému.
- Sečtěte charakteristiky a určete faktor technické složitosti systému.
- Určete počet upravených FP systému.

Dále v textu je uveden podrobnější popis jednotlivých činností v průběhu procesu počítání funkčních bodů.

#### 6.5.2.1 Počítání FTR, RET a DET

Pomocí počtu FTR (File Type Referenced), RET (Record Element Type) a DET (Data Element Type) vztažených k jednotlivým komponentám se určuje jejich složitost tak, jak je

uvedeno v následujících kapitolách, které popisují jednotlivé komponenty. Nejprve tedy definice jednotlivých pojmů:

- ***Record Element Type (RET)*** – Uživatelem rozpoznatelná podskupina datových elementů v ILF nebo EIF.
- ***File Type Referenced (FTR)*** – ILF nebo EIE odkazované pomocí transakce.
- ***Data Element Type (DET)*** – DET je unikátní uživatelsky rozeznatelné a nerekurzivní pole. Obsahuje dynamickou informaci čtenou ze souboru nebo FTR. Navíc se může odvolávat na transakci nebo může jít o další informace o transakcích. Jestliže jde o rekurzivní pole, pak je započítán jen první výskyt a ne každý výskyt.

Určení složitosti komponent pro výpočet funkčních bodů je závislé na počtu DET a RET nebo FTR tak, jak ukazuje následující tabulka.

| Komponenta | FTR | RET | DET |
|---|---|---|---|
| **External Input (EI)** | X | | x |
| **External Output (EO)** | X | | x |
| **External Inquiry (EI)** | X | | x |
| **Internal Logical File (ILF)** | | x | x |
| **External Interface File (EIF)** | | x | x |

**Tabulka 6-3 Vliv FTR, RET a DET na složitost komponent**

#### 6.5.2.2 Externí vstupy

Mezi externí vstupy (Extrnal Inputs - EI) započteme každá unikátní uživatelská data nebo zadání uživatelských povelů, která vstoupí přes externí rozhraní do aplikace a přidají, mění, ruší nebo jinak pozmění data (např. přiřazení, přemístění…) v interním logickém souboru. Započteme také řídící informaci, která vstoupí přes aplikační hranici a která zajistí soulad s funkcí specifikovanou uživatelem. Externí vstup by měl být považován za unikátní, pokud návrh vyžaduje logiku zpracování odlišnou od ostatních externích vstupů.

Zde jsou uvedeny příklady s ohodnocením:

- datová obrazovka s přidáním, změnou a rušením (3 EI)
- více obrazovek pohromadě zpracovaných jako jedna transakce (1 EI)

- dvě datové obrazovky s odlišným uspořádáním dat, ale se shodnou logikou zpracování (1 EI)
- dvě datové obrazovky se shodným formátem, ale s odlišnou logikou zpracování (2 EI)
- datová obrazovka s více unikátními funkcemi (1 EI za každou funkci)
- automatický vstup dat nebo transakcí z jiné aplikace (1 EI na každý typ transakce)

Následující tabulka znázorňuje určení složitosti externích vstupů. V závorce za složitostí je uveden počet funkčních bodů odpovídající externímu vstupu dané složitosti.

| **#FTR** | **#DET** | | |
|---|---|---|---|
| | **1-4** | **5-15** | **Více než 15** |
| **Méně než 2** | Malá (3) | Malá (3) | Střední (4) |
| **2** | Malá (3) | Střední (4) | Velká (6) |
| **Více než 2** | Střední (4) | Velká (6) | Velká (6) |

**Tabulka 6-4 Tabulka pro zjištění složitosti externích vstupů**

#### 6.5.2.3 Externí výstupy

Do externích výstupů (External Outputs - EO) započteme každá unikátní uživatelská data nebo řídící data, která opouští externí hranici měřeného systému. Externí výstup je považován za unikátní, pokud má odlišná data, nebo pokud vnější návrh (jiná aplikace) vyžaduje odlišnou logiku zpracování oproti jiným externím výstupům. Externí výstupy se často skládají z hlášení, výstupních souborů zasílaných jiné aplikaci nebo zpráv pro uživatele.

Zde jsou uvedeny příklady s ohodnocením:

- výstup dat na obrazovku (1 EO)
- souhrnná zpráva - dávkové zpracování (1 EO)
- automatická data nebo transakce směrem k jiným aplikacím (1 EO)
- chybové zprávy vrácené jako výsledek vstupní transakce (0 EO)
- záložní soubory (0 EO)
- výstup na obrazovku a na tiskárnu (2 EO)
- výstupní soubory vytvořené z technických důvodů (0 EO)
- výstup sloupcového a zároveň koláčového grafu (2 EO)

- dotaz s vypočtenou informací (1 EO, 0 EQ)

Následující tabulka znázorňuje určení složitosti externích výstupů. V závorce za složitostí je uveden počet funkčních bodů odpovídající externímu výstupu dané složitosti.

| #FTR | #DET | | |
|---|---|---|---|
| | **1-5** | **6-19** | **Více než 19** |
| **Méně než 2** | Malá (4) | Malá (4) | Střední (5) |
| **2-3** | Malá (4) | Střední (5) | Velká (7) |
| **Více než 3** | Střední (5) | Velká (7) | Velká (7) |

**Tabulka 6-5 Tabulka pro zjištění složitosti externích výstupů**

#### 6.5.2.4 Externí dotazy

Jako externí dotaz (External Inquires - EQ) započteme každou unikátní vstupně/výstupní kombinaci, kde vstup je příčinou a generuje výstup. Vnější dotaz je považován za unikátní, pokud se od ostatních dotazů odlišuje typem výstupních datových elementů, nebo pokud vyžaduje odlišnou logiku zpracování v porovnání s ostatními externími dotazy.

Zde jsou vedeny příklady s ohodnocením:

- on-line vstup a on-line výstup beze změny v datových souborech (1 EQ)
- dotaz následovaný změnovým vstupem (1 EQ/1 EI)
- vstup a výstup na obrazovce s nápovědou (na dané úrovni) (1 EQ)
- on-line vstup s bezprostředním tiskem dat bez jejich změny (1 EQ)
- výběr ze seznamu nebo z umístění s dynamickými daty (1 EQ)
- výběr ze seznamu nebo z umístění se statickými daty (0 EQ)
- požadavek na zprávu obsahující neodvozená data (1 EQ)

Následující tabulka znázorňuje určení složitosti externích dotazů. V závorce za složitostí je uveden počet funkčních bodů odpovídající externímu dotazu dané složitosti.

| #FTR | #DET | | |
|---|---|---|---|
| | **1-5** | **6-19** | **Více než 19** |
| **Méně než 2** | Malá (4) | Malá (4) | Střední (5) |
| **2-3** | Malá (4) | Střední (5) | Velká (7) |
| **Více než 3** | Střední (5) | Velká (7) | Velká (7) |

**Tabulka 6-6 Tabulka pro zjištění složitosti externích dotazů**

#### 6.5.2.5 Vnitřní logické soubory

Vnitřní logický soubor (Internal Logical File - ILF) je každá velká logická skupina uživatelských dat nebo informací použitých pro řízení aplikace. Do ILF zahrneme každý logický soubor, nebo v případě DB, každé logické seskupení dat z pohledu uživatele, které je vytvořeno, používáno, nebo udržováno aplikací. Spíše než fyzické soubory započteme každé logické seskupení dat tak, jak je viděno z pohledu uživatele, a jak je definováno při analýze požadavků nebo návrhu dat. Nezapočteme soubory, které nejsou přístupné uživateli prostřednictvím vnějšího výstupu nebo dotazu, a které nejsou nezávisle udržovány.

Zde jsou vedeny příklady s ohodnocením:

- logická entita nebo skupina entit z pohledu uživatele (1 ILF)
- logický interní soubor generovaný nebo udržovaný aplikací (1 ILF)
- uživatelem udržovaná tabulka nebo soubor (1 ILF)
- datový soubor nebo soubor s řídící informací, který aplikace použije při sekvenčním zpracování a údržbě (1 ILF)
- atributová entita udržovaná pouze prostřednictvím hlavní entity (0 ILF)
- asociativní entity vytvořené průnikem nebo spojením obsahující pouze klíčový atribut (0 ILF)
- přechodný nebo třídicí soubor (dočasný soubor) (0 ILF)
- soubor vytvořený proto, že byla použita určitá technologie (např. indexový soubor) (0 ILF)
- soubor s předlohou (vzorem), který aplikace pouze čte (0 ILF, 1 EIF)

Následující tabulka znázorňuje určení složitosti vnitřních logických souborů. V závorce za složitostí je uveden počet funkčních bodů odpovídající ILF dané složitosti.

| #RET | #DET | | |
|---|---|---|---|
| | **1-19** | **20-50** | **Více než 50** |
| **1** | Malá (7) | Malá (7) | Střední (10) |
| **2-5** | Malá (7) | Střední (10) | Velká (15) |
| **Více než 5** | Střední (10) | Velká (15) | Velká (15) |

**6-7 Tabulka pro zjištění složitosti vnitřních logických souborů**

#### 6.5.2.6 Soubory vnějšího rozhraní

Mezi soubory vnějšího rozhraní (External Interface Files - EIF) započteme každou velkou logickou skupinu uživatelských dat nebo řídící informace používané aplikací. Tato informace musí být udržována jinou aplikací. Zahrňte každý logický soubor nebo logickou skupinu dat z pohledu uživatele a každou velkou logickou skupinu uživatelských dat nebo řídící informace, která je extrahována aplikací z jiné aplikace ve formě souboru externího rozhraní. Extrakce nemá mít za následek změnu v některém z interních logických souborů. Pokud ano, pak započteme do EI místo do EIF.

Zde jsou vedeny příklady s ohodnocením:

- soubory nebo záznamy extrahované z jiné aplikace (použité pouze jako odkazy) (1 EIF)
- databáze čtená pomocí jiné aplikace (1 EIF)
- vnitřní logický soubor jiné aplikace použitý jako transakce (0 EIF, 1 EI)
- systém HELP, bezpečnostní soubor, chybový soubor čtený nebo odkazovaný aplikací, který pochází z jiné aplikace, která soubory udržuje (2 EIF)

Následující tabulka znázorňuje určení složitosti souborů vnějšího rozhraní. V závorce za složitostí je uveden počet funkčních bodů odpovídající EIF dané složitosti.

| #RET | #DET | | |
|---|---|---|---|
| | **1-19** | **20-50** | **Více než 50** |
| **1** | Malá (5) | Malá (5) | Střední (7) |
| **2-5** | Malá (5) | Střední (7) | Velká (10) |
| **Více než 5** | Střední (7) | Velká (10) | Velká (10) |

**6-8 Tabulka pro zjištění složitosti souborů vnějšího rozhraní**

### 6.5.2.7 Počítání upravených funkčních bodů

Neupravené funkční body se přepočítají na upravené funkční body neboli konečné ohodnocení velikosti projektu. K tomuto převodu se využívá ohodnocení pomocí 14 charakteristik hodnocených podle stupně vlivu na aplikaci. Každá charakteristika je ohodnocena číslem 0 – 5 podle toho, jak vystihuje vlastnosti projektu. Stupnice hodnocení je následující:

- 0 = bez vlivu
- 1 = náhodný
- 2 = mírný
- 3 = průměrný
- 4 = významný
- 5 = podstatný

Charakteristiky systému jsou pak následující:

1) Vyžaduje systém spolehlivé zálohování a zotavení?
2) Jsou vyžadovány datové komunikace?
3) Existuje distribuované zpracování?
4) Je výkonnost kritická?
5) Poběží systém v stávajícím intenzivně využívaném operačním prostředí?
6) Systém požaduje on-line vstup dat?
7) Vyžaduje on-line vstup dat použití vstupní transakce přes více obrazovek nebo operací?
8) Jsou hlavní soubory opravovány on-line?
9) Jsou vstupy, výstupy, soubory a dotazy složité?
10) Je vnitřní zpracování složité?
11) Je kód navrhován s cílem znovupoužití?
12) Jsou konverze a instalace zahrnuty v návrhu?
13) Je systém navrhován pro násobné instalace u různých organizací?
14) Je aplikace navrhovaná tak, aby zajistila změny a snadné používání na straně uživatele?

Výsledný faktor vlivu systémových charakteristik se vypočte podle následujícího vzorce:

$$VAF = 0{,}65 + \sum_{i=1}^{14} C_i/100$$

Kde $C_i$ je ohodnocení i-té charakteristiky.

Neupravené funkční body se pak převedou na upravené funkční body pomocí následujícího vzorce:

$$FP = UFP * VAF$$

#### 6.5.2.8 Rozdíl započítání FP u projektů na úpravu aplikace a tvorbu nové aplikace

Počítání FP pro projekty vytvářející nové aplikace je jasné. Spočteme jednoduše FP všech vlastností a dostaneme výsledek. Jak se ale zachovat u projektů, jejichž náplní je úprava stávající aplikace? Následující tabulka nám na to dá jasnou odpověď.

| Typ projektu | Projektové FP | Aplikační FP |
|---|---|---|
| **Tvorba nové aplikace** | Proj. FP = FP nových vlastností | App. FP = Nové FP |
| **Úprava stávající aplikace** | Proj. FP =<br>FP nových vlastností<br>+ FP odstraněných vlastností<br>+ FP změněných vlastností | App. FP =<br>FP nových vlastností<br>- FP odstraněných vlastností<br>+ ΔFP změněných vlastností |

**Tabulka 6-9 Způsob určení aplikačních FP, převzato z (Longstreet, 2001)**

### 6.5.3 Odhadování funkčních bodů

Uvedené metody odhadování funkčních bodů jsou doporučené pro odhadování ve fázích Project Definition a Solution Design. Všechny jsou podporované nástrojem PRESTO, který vás provede zadáváním dat a procesem odhadování. V tomto nástroji je rovněž možné vytvářet kombinované odhady. Co to je kombinovaný odhad se můžete dočíst v následující kapitole.

Doporučuji všechny metody popsané v této kapitole si podrobně prostudovat, abyste byli vždy schopni kvalifikovaně zvolit metodu odpovídající situaci a stavu projektu.

Pro studium této kapitoly se předpokládá znalost kapitoly 2 a 6.5.2.

#### 6.5.3.1 Katalog předpřipravených řešení

Tato metoda předpokládá vytvoření katalogu typových řešení, elementů, procesů atd. U těchto položek je uložena kromě podrobného popisu i jejich průměrná velikost v UFP. Nejlépe by měla být řešení rozdělena podle kompetenčního centra a velikosti projektu, aby se dosáhlo větší přesnosti.

Takový katalog bude v nástroji PRESTO postupně vznikat a v rámci podpory této metody je možné, jednotlivým úkolům v projektu nebo celým projektům, takováto předpřipravená řešení přiřadit.

#### 6.5.3.2 Průměrná složitost komponent

Tato metoda předpokládá to, že v projektech se vyskytují komponenty pro výpočet funkčních bodů s podobnou průměrnou složitostí. Tedy pokud spočteme průměrnou složitost komponent, můžeme pomocí toho odhadovat počet funkčních bodů, aniž bychom znali detailní vlastnosti komponent.

Nasazení této metody je vhodné, pokud jsme schopni spočítat jednotlivé komponenty, ale nejsme schopni určit jejich složitost. Případně je možné metodu zkombinovat s některou z metod v kapitole 6.5.3.3.

Příklad takové průměrné složitosti je zobrazen v následující tabulce, která je převzata z ISBSG benchmarku vývojových projektů.

| | **Průměrný počet UFP** |
|---|---|
| **ILF** | 7,4 |
| **EIF** | 5,5 |
| **EI** | 4,3 |
| **EO** | 5,4 |
| **EQ** | 3,8 |

**Tabulka 6-10 Průměrné složitosti komponent podle ISBSG benchmark**4

Počet neupravených funkčních bodů (Unadjusted Function Points - UFP) odhadneme následujícím vztahem:

$$Odh.UFP = \#EI * 4{,}3 + \#EO * 5{,}4 + \#EQ * 3{,}8 + \#ILF * 7{,}4 * \#EIF * 3{,}8$$

[4] www.isbsg.org

Tento obecný průměr může být při dostatku historických dat nahrazen koeficientem vlastním, počítaným z databáze.

#### 6.5.3.3 Extrapolace některých elementů pro výpočet FP na základě ostatních

Tato metoda předpokládá závislost mezi komponentami pro výpočet funkčních bodů. Takže v situaci, kdy nejsme schopni spočítat všechny komponenty z důvodu omezené znalosti řešení, je možné spočítat jen některé komponenty a ostatní dopočítat pomocí závislosti.

V literatuře existují již vyjádřené závislosti získané z obecných dat o softwarových projektech. Jsou to:

- **Tichenor ILF Model:**

$$Odh.UFP = \#ILF * 11{,}01 \ \ pro\ dávkové\ systémy$$

$$Odh.UFP = \#ILF * 14.93 \ \ pro\ transakční\ systémy$$

- **CNV AG:**

$$Odh.UFP = 7.3 * \#EO + 56$$

- **NESMA:**

$$Odh.UFP = 35 * \#ILF + 15 * \#EIF$$

- **ISBSG Benchmark:**

$$Odh.UFP = \frac{(7{,}4 * \#ILF)}{22} * 100$$

Toto jsou vztahy získané z obecných dat. V případě dostatku historických dat budou vypočítány vlastní koeficienty a bude umožněno jejich používání.

### 6.5.4 Kombinované odhady

Kombinovaným odhadem rozumíme to, když se odhad na fázi nebo celý projekt provede několika různými metodami. Tyto odhady se následně zkombinují a vznikne

kombinovaný odhad. Kombinovaný odhad je spočítán například zprůměrováním výsledků použitých metod.

Nástroj PRESTO tyto kombinované odhady podporuje, a pokud je odhad proveden pomocí více metod, nástroj automaticky počítá kombinovaný odhad a prezentuje jeho výsledky.

# 7 Nástroj pro podporu odhadování

Project Effort Estimation Tool (dále jen PRESTO) je nástroj pro podporu odhadování softwarových projektů.

Základem nástroje je databáze, která obsahuje historická data z projektů a další data používaná pro kalibraci odhadovacích metod a kalkulaci odhadů. Struktura této databáze je dále popsána v kapitole 7.1.1.1.

## 7.1 Specifikace nástroje

V této kapitole je uvedena podrobnější specifikace nástroje PRESTO. V jednotlivých kapitolách popisuji databázi, se kterou bude nástroj pracovat, tedy hlavně popis dat, která v ní budou uložena. Dále popis jednotlivých modulů sloužících pro zakládání projektů, následné zadávání dat o jejich průběhu, přehled skutečně spotřebovaných zdrojů a nákladů, tvorbu a počítání odhadů a prezentaci výsledků a srovnání projektů.

### 7.1.1 Databáze

Databáze obsahuje data z projektů, a to jak z těch běžících, tak z těch již uzavřených. Obsahuje obecná data a parametry popsané v následující kapitole, které se využívají při tvorbě odhadu. V neposlední řadě jsou zde uvedeny generované statistiky z dat o projektech. Databáze obsahuje také předpřipravená řešení s pracností a ohodnocením velikosti.

#### 7.1.1.1 Struktura databáze

Databáze obsahuje následující data.

**Data z projektů:**

- obecné informace o projektu
    - jméno projektového manažera (IT PL)
    - jméno projektu
    - stručný popis projektu
    - velikost projektu ve funkčních bodech
    - typ projektu (HW, SW, proces)
    - kompetenční centrum

    - druh projektu integrační, implementační, nová aplikace, úprava stávající aplikace
- odhadované časové vymezení jednotlivých fází a náklady
    - odhadované doby trvání jednotlivých fází pro odhady v jednotlivých fázích projektu
    - odhadované náklady a počty MD na řešení
- skutečné časové vymezení jednotlivých fází
    - data schválení jednotlivých fází
    - datum nasazení projektu
    - datum ukončení projektu
    - doba trvání jednotlivých fází
- skutečné náklady a počty MD na jednotlivé fáze
    - náklady a počty MD na jednotlivé fáze rozdělené podle pracovníků na interní a externí a dále podle zaměření na analýzu, vývoj, testy, podporu a management
    - náklady na HW
- parametry z odhadovacích metod
    - parametry nastavení metod při odhadování v jednotlivých fázích
    - parametry pro tvorbu odhadu ve fázi IF

**Generované statistiky z projektových dat:**

- poměry rozdělení nákladů mezi interní, externí zdroje a mezi analýzu, vývoj, testy, management, podporu …
- koeficienty určené pro odhadovací metody

**Obecná data:**

- cena za MD (analýza, vývoj, testy, ITPL)
    - interní
    - externí
- cena za HW přes všechny kategorie dostupnosti a platformu
    - databázový server
    - aplikační server
    - web server
    - uživatelské PC

    - ostatní HW
- informace o předpřipravených řešeních pro využití v odhadech
    - popisy jednotlivých předpřipravených řešení
    - náklady na jednotlivá řešení

### 7.1.2 Sběr dat

Nástroj umožňuje administrovat jednotlivé projekty v databázi. Je tedy možné je zakládat a upravovat.

### 7.1.3 Odhadování

Pro podporu odhadování je v nástroji algoritmicky zpracovaná podpora pro metodiku, která je podrobně popsána v kapitole 6 této práce.

Pro každý založený projekt je umožněn přístup k podpoře odhadování. Nástroj obsahuje podporu pro všechny odhadovací metody popsané v metodice. Pro všechny metody jsou v rámci tohoto modulu vypočítávány koeficienty pro jejich nastavení.

Podpora nástroje poskytovaná odhalovateli spočívá v automatické kalibraci odhadovacích metod vypočítanými koeficienty a vlastní odhad pak probíhá pomocí formuláře. Odhadovatel v něm postupně nastaví jednotlivé parametry odhadovaného projektu požadované danou metodou. Nástroj následně provede kalkulaci odhadu.

Odhady pro jednotlivé fáze jsou ukládány do databáze pro možné pozdější zobrazení a případnou konfrontaci odhadu se skutečností.

### 7.1.4 Prezentace odhadů

Nástroj umožňuje prezentaci odhadů, a to jak ve formě číselných výpisů, tak ve formě vizualizace pomocí grafu. Je zde možnost srovnávat čerpané a odhadované náklady a také možnost zobrazit statistiky projektu a srovnání se statistikami globálními.

# 8 Případová studie

Dále v textu uvádím případovou studii odhadování pomocí navržené metodiky. Studie je provedena na třech již ukončených projektech z KB.

Jak je napsáno v kapitole 6.4.4, pro převod odhadnuté velikosti funkčních bodů na práci bylo použito koeficientu získaného z obecných dat. Historická data z Komerční banky obsahující náklady a práci potřebnou na dokončení projektů byla využita pouze při odhadování ve fázi Idea Formulation.

## 8.1 Projekt první

Případová studie prvního projektu.

### 8.1.1 Odhady

V tabulce 8-1 jsou uvedeny odhady projektu provedené v jednotlivých fázích a přepočet odhadnutých funkčních bodů na MD práce zaměstnanců potřebných pro dokončení projektu.

| Projektová fáze | Metoda odhadování | Odhadnutá velikost ve funkčních bodech | Odhadnutý počet MD potřebný na dokončení projektu |
|---|---|---|---|
| **Idea Formulation** | Odhad ve fázi Idea Formulation | n | 2000 |
| **Project Definition** | NESMA | 1567 | 2208 |
| **Solution Design** | Počítání funkčních bodů | 1234 | 1771 |

**Tabulka 8-1 Odhady jednotlivých fází projektu**

### 8.1.2 Srovnání s realitou

V tabulce 8-2 je uvedeno srovnání odhadů v jednotlivých fázích s reálně spotřebovanými MD na projektu. Je zde také zobrazena procentuální odchylka odhadu od reality.

| Projektová fáze | Odhadnutý počet MD potřebný na dokončení projektu | Reálně spotřebovaný počet MD | Odchylka odhadu v procentech |
|---|---|---|---|
| **Idea Formulation** | 2000 | 1523 | +31% |
| **Project Definition** | 2208 | 1523 | +45% |
| **Solution Design** | 1771 | 1523 | +16% |

**Tabulka 8-2 Srovnání odhadů s realitou**

## 8.2 Projekt druhý

Případová studie druhého projektu.

### 8.2.1 Odhady

V tabulce 8-3 jsou uvedeny odhady projektu provedené v jednotlivých fázích a přepočet odhadnutých funkčních bodů na MD práce zaměstnanců potřebných pro dokončení projektu.

| Projektová fáze | Metoda odhadování | Odhadnutá velikost ve funkčních bodech | Odhadnutý počet MD potřebný na dokončení projektu |
|---|---|---|---|
| **Idea Formulation** | Odhad ve fázi Idea Formulation | n | 1000 |
| **Project Definition** | ISBSG | 789 | 1159 |
| **Solution Design** | Počítání funkčních bodů | 430 | 644 |

**Tabulka 8-3 Odhady jednotlivých fází projektu**

### 8.2.2 Srovnání s realitou

V tabulce 8-4 je uvedeno srovnání odhadů v jednotlivých fázích s reálně spotřebovanými MD na projektu. Je zde také uvedena procentuální odchylka odhadu od reality.

| Projektová fáze | Odhadnutý počet MD potřebný na dokončení projektu | Reálně spotřebovaný počet MD | Odchylka odhadu v procentech |
|---|---|---|---|
| **Idea Formulation** | 1000 | 776 | +29% |
| **Project Definition** | 1159 | 776 | +49% |
| **Solution Design** | 644 | 776 | -17% |

**Tabulka 8-4 Srovnání odhadů s realitou**

## 8.3 Projekt třetí

Případová studie třetího projektu.

### 8.3.1 Odhady

V tabulce 8-5 jsou uvedeny odhady projektu provedené v jednotlivých fázích a přepočet odhadnutých funkčních bodů na MD práce zaměstnanců potřebných pro dokončení projektu.

| Projektová fáze | Metoda odhadování | Odhadnutá velikost ve funkčních bodech | Odhadnutý počet MD potřebný na dokončení projektu |
|---|---|---|---|
| **Idea Formulation** | Odhad ve fázi Idea Formulation | n | 6000 |
| **Project Definition** | NESMA | 3546 | 4461 |
| **Solution Design** | Počítání funkčních bodů | 5450 | 6062 |

**Tabulka 8-5 Odhady jednotlivých fází projektu**

### 8.3.2 Srovnání s realitou

V tabulce 8-6 je uvedeno srovnání odhadů v jednotlivých fázích s reálně spotřebovanými MD na projektu. Je zde také uvedena procentuální odchylka odhadu od reality.

| Projektová fáze | Odhadnutý počet MD potřebný na dokončení projektu | Reálně spotřebovaný počet MD | Odchylka odhadu v procentech |
|---|---|---|---|
| **Idea Formulation** | 6000 | 5819 | +3% |
| **Project Definition** | 4461 | 5819 | -23% |
| **Solution Design** | 6345 | 6624 | +14% |

**Tabulka 8-6 Srovnání odhadů s realitou**

# 9 Závěr

Odhadování softwarových projektů je komplikovaná disciplína softwarového inženýrství, která může někomu připadat jako věštění z křišťálové koule. Doufám, že tato práce přispěla k objasnění některých problémů spojených s odhadováním projektů a přiblížila čtenáři několik základních způsobů odhadování a odhadovacích metod.

V rámci práce jsem vypracoval metodiku na odhadování softwarových projektů, která má primárně sloužit pro potřeby IT oddělení Komerční banky. Tato metodika standardizuje postup odhadování a zavádí jednotný systém sběru a údržby dat jednotlivých projektů. Je založena na odhadování velkosti projektů ve funkčních bodech a používá pro to vhodné odhadovací metody. Pro každou fázi jsou určeny metody odpovídající konkrétnímu stupni rozpracovanosti zadání a požadované přesnosti. Metodika je stavěna tak, aby reflektovala konkrétní potřeby Komerční banky zjištěné v provedené SWOT analýze. Poněvadž jsou požadavky na odhadování a projektové řízení v jiných společnostech podobné, je jistě použitelná i jinde než jen v Komerční bance. Vytvořil jsem tedy universální metodiku na odhadování projektů. Navíc je metodika a s ní i všechny odhadovací metody dostatečně uživatelsky srozumitelná. Splňuje tak zadání ve všech jeho bodech.

Co se týče přesnosti metodiky, tak ta se dle provedené studie pohybuje na hranici požadovaných mezí. To je způsobeno použitím obecných dat pro kalibraci metod. Pokud bychom měli použitelná data z Komerční banky, byly by výsledky zajisté mnohem lepší. Přesnost metodiky se s postupem času a přibývajícími daty bude dále zlepšovat.

Dalším cílem práce bylo vytvoření jednoduchého nástroje pro podporu odhadování. Vytvořený nástroj kompletně podporuje celou metodiku a tvorbu odhadu. Navíc obsahuje databázi historických dat o projektech. Nabízí tedy řešení sběru a údržby dat o projektech v rámci společnosti. V nástroji je také možné zobrazovat reporty z běžících i skončených projektů. I zde se mi tedy podařilo cíl práce, tedy vytvořit jednoduchý nástroj pro podporu odhadování, naplnit.

Vzhledem ke skutečnostem popsaným v textu, nebylo možné pro odhadování pomocí navržené metodiky použít data z projektů KB. Proto je zde veliký potenciál pro zlepšení odhadovacích schopností metodiky. Toto bude probíhat s postupem času a přibývajícími vlastními daty. U jednotlivých odhadovacích metod je též možnost pro použití historických dat a pro zpřesnění výsledků. Navíc bude důležité sledovat přesnost odhadování a v případě větších rozdílů mezi odhadovanými a reálnými daty metodiku upravit. Je nutné metodiku

chápat jako dynamický organismus a hlavně z počátku její existence ji stále sledovat a případně modifikovat.

Nástroj pro podporu odhadování vytvořený v rámci této práce je dostatečně komplexní. Je ale jistě možné ho dále rozvíjet. Například přidávat další metody pro odhadování v jednotlivých fázích. Také by bylo zajímavé rozpracovat možnosti kapacitního plánování nástroje, aby byli uživatelé schopni sledovat a optimalizovat mapování projektů na dostupné kapacity. To jsou však již zlepšení přesahující rámec této práce.

# Použitá literatura

**Cohn, Mike. 2005.** Estimating With Use Case Points. *METHODS & TOOLS.* 2005, (Volume 13 - number 3).

**Coombs, Paul. 2003.** *IT Project Estimation: Guide to the Costing of Software.* s.l. : Cambridge University Press, 2003.

**Grady, Robert B. and Caswell, Deborah L. 1987.** *Software Metrics: Establishing a Company-Wide Program.* s.l. : Prentice-Hall, Inc., 1987. **ISBN 0-13-821844-7.**

**KB a.s. 2008.** *Projektová metodika KB a.s.* [Dokument] Praha : KB a.s., 2008.

**Longstreet, David. 2001.** *Function Points Analysis Training Course.* [Dokument] : Longstreet Consulting Inc, www.SoftwareMetrics.Com, 2001.

**McConnell, Steve. 2006.** *Odhadování softwarových projektů.* Brno : Computer Press, a.s., 2006. ISBN 80-251-1240-3.

**Meli, Roberto a Santillo, Luca. 1999.** Function Point Estimation Methods: a Comparative Overview. [Online] 1999. www.dpo.it/resources/papers/1999-fesma-fpestmet-en.pdf.

**Meli, Roberto. 1997.** *Early and Extended Function Point:.* Scottsdale, Arizona USA : IFPUG - Fall Conference, 1997.

**Rico, David F. 2004.** *ROI of Software Proces Improvement: Metrics of Project Managers and Engineers.* : J. Ross Publishing, Inc., 2004.

**Themis, James. 1997.** METHODS AND MODELS FOR ESTIMATING A SOFTWARE PROJECT: An Analytical Approach. [Online] 1997. http://www2.umassd.edu/SWPI/costmodeling/papers/methods-and-models-for.pdf.

# Přílohy

Na CD přiloženém k práci se nachází následující přílohy:

- Text této práce v elektronické podobě (cesta k dokumentu na CD je \prilohy\DP_OdhadovaniProjektu.pdf)
- Dokumentace k nástroji PRESTO (cesta k dokumentu na CD je \prilohy\Dokumentace_PRESTO.pdf)
- Kód a instalace nástroje PRESTO (na CD v adresáři \PRESTO\)