Regresní model pro odhadování nákladů

Bc. Dagmar Janečková

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Dagmar Janečková**
Osobní číslo:	**A14402**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**prezenční**
Téma práce:	**Regresní model pro odhadování nákladů**
Téma anglicky:	**Cost Estimation Regression Model**

Zásady pro vypracování:

1. **Seznamte se s principy a možnostmi regresních modelů.**
2. **Popište algoritmické metody odhadování úsilí při vývoji softwarových projektů.**
3. **Vybranou metodu odhadování optimalizujte pomocí vybraných regresních modelů.**
4. **Porovnejte dosažené výsledky vybraných regresních modelů.**
5. **Proveďte vyhodnocení přínosu aplikace regresních modelů pro odhadování.**
6. **Navrhněte další možný rozvoj s cílem zpřesnění výsledků odhadování.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MCCONNELL, Steve. Odhadování softwarových projektů: jak správně určit rozpočet, termín a zdroje. Brno: Computer Press, 2006. ISBN 80-251-1240-3.**
2. **TRENDOWICZ, Adam a Ross JEFFERY. Software project effort estimation: foundations and best practice guidelines for success. Switzerland: Springer International Publishing, 2014. ISBN 978-3-319-03628-1. ISBN 978-3-319-03629-8 (eBook).**
3. **MENDES, Emilia. Cost estimation techniques for web projects. Hershey PA: IGI Pub., c2008. ISBN 978-159-9041-377. ISBN 978-1-59904-137-7 (eBook).**
4. **GALORATH, Daniel D. a Michael W. EVANS. : when performance is measured performance improves. Boca Raton, FL: Auerbach Publications, 2006. ISBN 08-493-3593-0.**
5. **KOIRALA, Shivprasad. How to prepare Software Quotation [online]. Dostupné z: http://webapps.6te.net/HowtoPrepareSoftwareQuotations.pdf**
6. **SILHAVY, Radek; SILHAVY, Petr; PROKOPOVA, Zdenka. Analysis and selection of a regression model for the Use Case Points method using a stepwise approach. Journal of Systems and Software, 2017, 125: 1-14.**
7. **SILHAVY, Radek; SILHAVY, Petr; PROKOPOVA, Zdenka. Algorithmic optimisation method for improving Use Case Points estimation. PloS one, 2015, 10.11: e0141887.**

Vedoucí diplomové práce: **Ing. Radek Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **17. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

Ing. Miroslav Matýsek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

ABSTRAKT

Diplomová práce se zabývá problematikou odhadováním nákladů při vývoji softwarových projektů. V první části práce je čtenář uveden do problematiky tvorby odhadů a seznámen se základními pojmy jako je odhad, kvalita odhadu, chyby při odhadech, aj. Poté je čtenář seznámen se základními metodami pro tvorbu odhadů a s regresními modely.

Cílem práce je navrhnout vhodný regresní model k dosažení co nejlepších výsledků pro odhadování nákladů metodou Use Case Points, se zaměřením se na počet lidských hodin práce strávené na vývoji projektu.

Klíčová slova: metody odhadování, odhad, regresní model, UCP

ABSTRACT

This thesis is focused on issues of the calculation of costs during developments of software projects. In the first part of the thesis the reader is introduced into problems of formations of estimates and familiarized with terms such as estimates, a quality of estimates, errors in estimates, etc. Then the reader is familiarized with the basic methods for the production of estimates and with regression models.

The aim of the thesis is to design a suitable regression model to achieve the best results for estimating costs by the method Use Case Points, focusing on the number of human hours spent on the development of the project.

Keywords: estimation methods, estimate, regression model, UCP

Ráda bych poděkovala vedoucímu diplomové práce Ing. Radkovi Šilhavému, Ph.D. za odborné vedení, cenné rady a pomoc při tvorbě diplomové práce.

OBSAH

ÚVOD

V softwarovém inženýrství vycházejí odhady nákladů a délky vývoje z předpovědi velikosti projektu, pro který je odhad vytvořen. Na začátku projektu je však, díky nedostatku podrobných informací, vždy těžké spolehlivé odhady získat. Předběžné odhady na začátku projektu obsahují mnoho prvků nejistoty, kterým se v rané fázi nelze vyhnout. Přesto jsou tyto předběžné odhady vyžadovány při výběrovém řízení nebo při vyhodnocování, zda je daný projekt proveditelný. Je však důležité uvědomit si, že taková předpověď je užitečná pouze tehdy, je-li dostatečně přesná. Pro docílení zlepšení těchto předběžných odhadů se organizace často spoléhají na své předchozí zkušenosti a vychází z informací získaných z již hotových projektů. Takto reálně vidí, jaké nečekané problémy se v počátku vývoje ukazovaly, jak jim co dlouho trvalo a přesněji tedy tyto parametry do odhadu nového projektu započítají.

V teoretické části diplomové práce bude čtenář seznámen se základní terminologií odhadování. Dozví se, jaký je rozdíl mezi odhadem a plánováním a jaké vlivy mohou působit na tvorbu odhadu. Poté bude uveden základní přehled metod k odhadování, které se používají. Praktická část bude pracovat s algoritmickými metodami, bude jim proto v teoretické části věnováno více prostoru. Poslední kapitola teoretické části bude věnována popisu lineární a krokové regrese, kde si nejprve matematicky popíšeme lineární regresi a poté si vysvětlíme, jak prakticky funguje krok po kroku.

Praktickou část otevřeme popisem cílů práce, kde si přesně stanovíme, jakým směrem se bude další část práce ubírat. Hlavním cílem této práce je navrhnout regresní model pro odhadování nákladů. K dispozici jsou zdrojová data, na kterých bude postavena celá tato část. K docílení otestování vytvořených regresních modelů na nezávislých datech, budou zdrojová data na začátku práce rozdělena na dvě části. Na první části poté proběhne vytvoření regresních modelů, na druhé části pak jejich otestování a následné vyhodnocení.

Regresní analýza se využívá v situacích, kdy chceme znát závislost určité závislé proměnné na jedné nebo více nezávislých proměnných. Závislou proměnnou rozumíme proměnnou vysvětlovanou, kterou chceme danou regresí popsat a nezávislou proměnnou se rozumí vysvětlující proměnná, tedy taková, která k popisu přispívá. Počet těchto nezávislých proměnných značí, zda se bude jednat o model jednoduché regrese nebo vícenásobné regrese.

I. TEORETICKÁ ČÁST

1 PRINCIPY ODHADOVÁNÍ PŘI VÝVOJI SOFTWARE

Při vývoji software se odhaduje především velikost projektu, doba vývoje projektu a celkové náklady na vývoj. Je důležité chápat, jak úzce spolu všechny tyto údaje souvisí. Na vývoji software pracuje daný tým pracovníků, kterým je potřeba za jejich práci zaplatit. Čím větší projekt je, tím více času na něm pracovníci stráví a tím vyšší mzdu za odvedenou práci dostanou. Právě mzdy pracovníků tvoří největší část celkových nákladů. Většina metod pro tvorbu odhadů odhaduje především úsilí, které vyjadřuje počet potřebných člověkoměsíců pro odhadovaný projekt. Počtem člověkoměsíců se rozumí počet odpracovaných hodin na jednoho pracovníka za jeden měsíc.

V této kapitole si vysvětlíme, co to vlastně odhad je a co ovlivňuje jeho kvalitu.

1.1 Co je to odhad

Odhad lze definovat mnoha způsoby. Může se jednat o úsudek postavený na dojmech a názorech jednotlivce, o prozatímní vyhodnocení projektu, může jít o hrubou či předběžnou kalkulaci projektu, aj. Stavět odhad na malém vzorku dat, bez jasných představ o cíli projektu a bez použití jakékoli kvantitativní metody odhadu, je však velmi obtížné.

Pojem „odhad“ je lidmi často vykládán chybně. Cílem této kapitoly je vysvětlit, co vlastně odhad znamená a jak se liší od cílů, závazků a plánů.

1.1.1 Odhady, cíle a závazky

Odhad [1] je tedy vlastně předpověď na otázky jako např. jak dlouho bude projekt trvat a jaká bude jeho cena. Je důležité si ale uvědomit, že toto odhadování je navzájem ovlivňováno s obchodními cíli, závazky a taky řízením.

Cíl je to, čeho chceme dosáhnout [1]. Tedy konkrétní podmínky, jako např. stavení maximální částky rozpočtu pro vývoj, stanovení termínu pro dokončení projektu, aj. Tyto cíle ovšem nastavují ti, kteří mají mnohdy nerealistické očekávání, a tudíž nejsou vždy dosažitelné. Cíl je tedy popis žádoucího obchodního plánu, ale bez ohledu na odhad software.

Závazek je slib, schopnost k určitému datu dodat projekt na dané úrovni kvality [1]. Závazek může být stejný jako odhad, není to ale pravidlem, proto je potřeba jednotlivé pojmy nezaměňovat.

1.1.2 Odhady a plány

Zatímco odhadování je nezaujatý a analytický proces, u kterého se klade důraz především na přesné stanovení předpokládané doby trvání projektu nebo výše nákladů, plánování je již zaujatý proces, jehož cílem je hledání konkrétního výsledku. U plánování tedy dochází k vhodnému upravování plánů za účelem dosažení daného výstupu. Plánují se konkrétní způsoby, jak stanoveného cíle dosáhnout. Stanovené odhady jsou základem pro tvorbu plánu, tento plán se ale nemusí s odhadem vždy shodovat. V případě, že je odhad a cíl značně odlišný, musí plán tento fakt odhalit, vzít na vědomí a toto vyšší riziko započítat.

Obě tyto činnosti, odhadování i plánování, jsou důležité, nicméně jejich vzájemné odlišnosti znamenají, že jejich míchání má sklon vést ke špatným odhadům a následně i ke špatným plánům. V případě nahrazení analyticky vypočteného odhadu pouhým cílem v plánování mohou členové projektu na tento cíl odkazovat stejně jako na odhad, čímž odhad přestává být objektivní.

Největším problémem při odhadování a plánování bývá komunikace mezi jednotlivými stranami. Typickým příkladem je situace, kdy nadřízený požaduje vytvořit projekt a vymezí své, mnohdy nereálné požadavky, jako lhůtu na dokončení, kolik vývojářů může tomuto projektu přidělit aj. Vedoucí projektu si zadaný úkol s požadavky projde a přijde s odhadem, že tento projekt nejsou schopni dokončit ve stanoveném časovém horizontu. A v tomto bodě mnohdy nastává problém v komunikaci. Nadřízený požaduje, aby projekt stihli v něm stanoveném termínu. Zde je vidět, že nadřízený vlastně nechtěl pro vedoucím projektu vytvořit odhad, ale vypracovat plán, kterém by se cíle dalo dosáhnout. Toho nicméně vedoucí projektu není schopen. Správná komunikace by měla pokračovat tak, kdy se nadřízený zeptá vedoucího projektu i na jeho názor, jak realistické je splnění zadaného úkolu a následné ujasnění si priorit, zda je důležitější 100% funkčnost nebo alespoň nějaký funkční výsledek do požadovaného data. Po této domluvě může vedoucí projektu navrhnout takový plán, který bude přesně korespondovat s potřebami daného projektu.

Přesně stanovené odhady [1] jsou základem správného plánování k dosažení stanoveného cíle. Kvalita odhadu bude popsána v další části práce.

1.1.3 Definice „dobrého“ odhadu

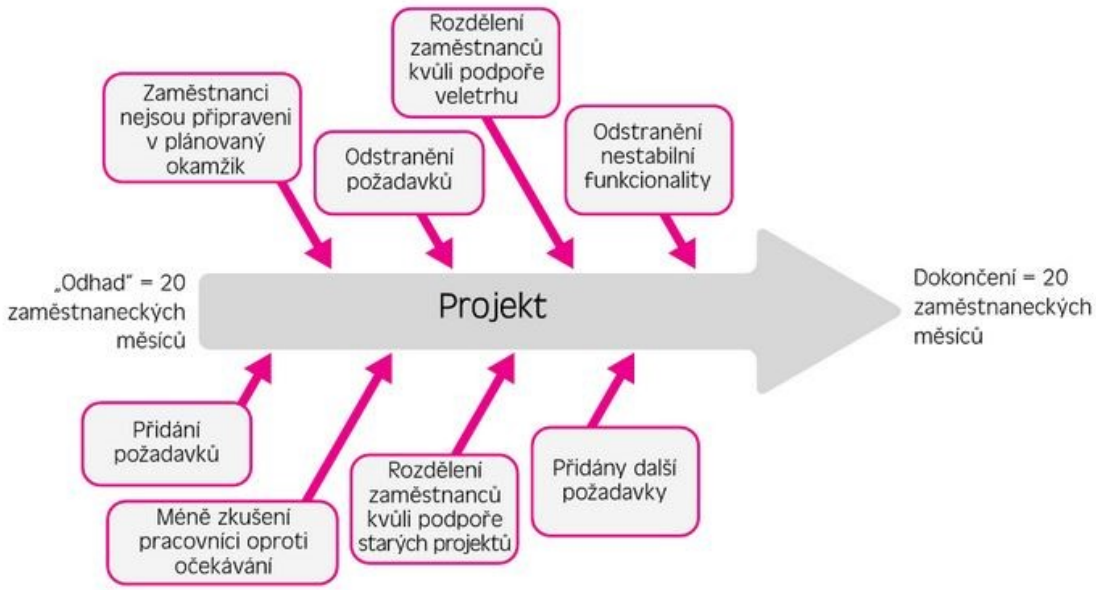
I když jsme si již definovali, co je to odhad, není to odpověď na otázku, co je to *dobrý* odhad. Na tuto otázku bylo experty na odhady navrženo mnoho definic. Capers Jones definoval [2]

přesnost dobrého odhadu ±10%, ale jen u dobře řízených projektů. Chaotické projekty díky příliš velké variabilitě takovéto přesnosti dosáhnout nemohou.

Nejběžnější standard pro vyjadřování přesnosti odhadů navrhli v roce 1986 profesoři S. D. Conte, H. E. Dunsmore a Y. D. Shen [3]. Standard se zakládá na tom, že opravdu dobrým přístupem k odhadování by se měly dát získat odhady v rozmezí 25% okolo skutečných výsledků v 75% času.

Tento přístup znamená postupné zlepšování přesnosti odhadů v průběhu času. Odhad stanovený na začátku projektu nedosahuje takové přesnosti, jaké je schopen dosáhnout v dalších etapách, úrovních. K tomu zpřesnění pomáhají především historická data z dané společnosti, z dané problematiky.

V době, kdy byl tento přístup navržen, spousta společností, která vytvářela odhady na základě výše zmíněných definic, ohlásila, že dosahují výsledků odhadů, které jsou blízko ke stanovené přesnosti. Některé společnosti prohlásily, že jsou schopny plnit své závazky včas a za smluvenou cenu v 97%, některé dokonce i v 98%. Nicméně je nezbytné zmínit, že obě definice neobsahují důležitý koncept, a sice že přesných výsledků nelze dosáhnout pouze praktikováním odhadů, ale tyto odhady musí být podepřeny efektivním řízením daného projektu.



Obr. 1 - Řízení a odhad projektu [1, str. 34]

Jelikož se projekty od jejich počátků až do dokončení mění, výsledný projekt nebývá projekt, pro který byl odhad vytvořen. Jak lze vidět na obrázku výše (Obr. 1), v průběhu procesu

dochází ke změnám jako odstraňování nedůležitých požadavků, změna definování některých požadavků, přidávání požadavků, nedostatek kvalifikovaných pracovníků atd. Pokud ale dokončení projektu odpovídá termínu dokončení stanoveném v odhadu, lze říci, že projekt souhlasí s odhadem.

Kritéria dobrého odhadu tedy nemohou být založena na jeho schopnosti předvídat, ale na jeho schopnosti podpořit úspěch projektu [1].

1.1.4 Skutečný cíl odhadů

Ti, kteří mají za úkol vypracovat odhad nějakého projektu, stojí často před odlišností mezi obchodními cíli a odhadovaným časem a náklady. V lepším případě, kdy je tento rozdíl nepatrný, lze díky pečlivé případě či drobnou úpravou časového rozvrhu, rozpočtu nebo vlastností projektu, dostát svým závazkům a dovést projekt ke zdárnému konci. Jsou-li ale rozdíly mezi cíli a odhadem hodně odlišné, je potřeba přehodnotit cíl projektu. Hlavním cílem odhadů tedy není předpovědět dokončení daného projektu, ale zjištění, zda jsou obchodní cíle realistické, a tedy zda je projekt zvládnutelný.

V praxi bylo zjištěno, že pokud se počáteční cíl a odhad od sebe neliší o více než 20%, tak má projektový manažer dostatek prostoru pro řízení funkčnosti projektu, rozdělení práce a dalších nezbytných parametrů [1].

Pracovní definice „dobrého“ odhadu zní: *„Dobrý odhad je odhad, který poskytuje dostatečně jasný pohled na realitu projektu, aby vedení projektu mohlo dělat dobrá rozhodnutí, jak projekt vést, aby bylo dosaženo cíle.“* [1, str. 36]

1.2 Kvalita odhadu

Jedním z důvodů pro vytváření neúspěšných odhadů je nedostatečná znalost pozadí v oblasti odhadu software. Možnost libovolného využití metod a nástrojů pro odhad často vedou k neuspokojivým výsledkům, zatímco hlavní příčiny zůstávají nejasné [4].

Nízká kvalita odhadů softwarových projektů, jejich nesplnitelné cíle a závazky, jsou problémem už spoustu let. Na selhání softwarových projektů díky nedostatku času upozornil již v roce 1975 Fred Brooks [5], o devět let později Scott Costello přišel se svým postřehem [6], že „tlak termínu je jednoduše největší nepřítel vývoje softwaru“. O dalších deset let, v roce 1994 prohlásil Caper Jones, že „nepřiměřené nebo iracionální plány jsou pravděpodobně

nejdestruktivnějším vlivem ve všem, co se týká software“. Přesně stanovený odhad je tedy vzácnost. V této kapitole budou postupně rozebrány jednotlivé aspekty kvalitního odhadu.

1.2.1 Přesnost odhadu

Při zpracovávání odhadu softwarového projektu se nabízí celkem logická otázka – je lepší odhad nadhodnotit nebo podhodnotit? Přesný odhad je základem pro plánování celého projektu. Díky přesnému odhadu je schopen vedoucí vývoje efektivně rozdělit práci mezi jednotlivé vývojové oddělení, naplánovat časové termíny dodávek určitých částí projektu s přesností na den, hodinu, klidně i minutu. Nicméně takto přesné odhady jsou v praxi velmi vzácné, proto je při odhadování důležité rozhodnutí, zda je lepší nadhodnocení nebo podhodnocení.

Nadhodnocení projektu s sebou může nést jistá úskalí. Pokud je na dokončení nějakého projektu stanoven odhad 6 měsíců, přičemž lze úkol splnit za kratší dobu, vývojář si vždy najde dostatek další práce, kterou může ve zbývajícím čase vykonávat. Jedná se o Parkinsonův zákon – vždy se najde dostatek činnosti k využití celého času. Díky tomuto se manažeři a investoři projektů snaží nadhodnocování spíše vyhnout, jelikož to přináší i zbytečné výdaje. Kromě Parkinsonova zákona existuje ještě Goldrattův „studentský syndrom“, kde pokud mají vývojáři spoustu času a práci odkládají. V určitém okamžiku, kdy začnou intenzivně pracovat, už není jisté, zda projekt v termínu stihnou.

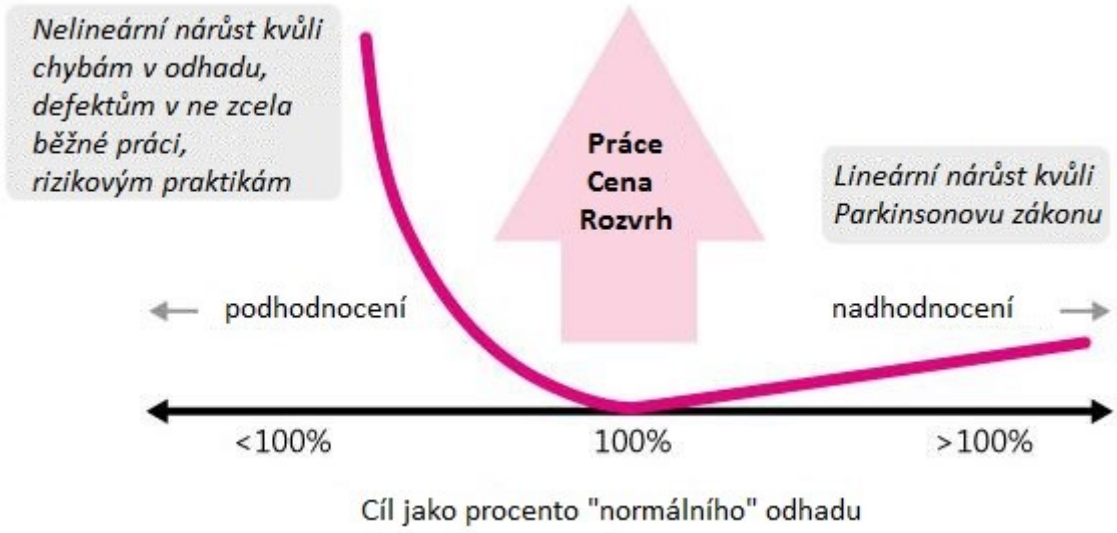
Podhodnocení projektu s sebou přináší, kromě snahy docílit u vývojového týmu pocit naléhavosti, i mnoho problémů [1]:

1. *Redukovaná efektivita plánů projektu* – podhodnocené odhady znemožňují efektivní rozplánování projektu, např. nedostatečný počet vývojářů, špatná koordinace mezi jednotlivými odděleními, aj. Chyba v odhadu v rozmezí 5 % až 10 % by nezpůsobila žádné větší problémy, avšak tyto chyby jsou často nepřesné o více než 100 %. Vzhledem k takové nepřesnosti jsou zhotovené plány v podstatě k ničemu.
2. *Statisticky redukovaná šance na včasné dokončení* – při odhadování vývojáři odhadují o 20 % až 30 % méně, než je jejich skutečná práce. Požadavky na projekt se totiž v průběhu času mění a nelze předpovědět všechny podmínky a situace, které mohou nastat. Zredukování základního odhadu bychom tedy ještě zmenšili šanci na včasné dokončení.
3. *Špatné technické základy vedou k horším výsledkům, než je běžné.* V případě, že vývojář nedostane dostatečný časový prostor pro aktivity, jako požadavky nebo design

projektu, bude se k nim muset vrátit v budoucnu. Přináší to s sebou kromě vyšší ceny za vývoj projektu i delší čas vývoje, než kdyby se vycházelo z přesného odhadu.

4. *Destruktivní dynamika ve zpožďujícím se projektu jej zhorší oproti normálu*. Ve chvíli, kdy je projekt oproti plánu ve zpoždění, je projektový tým zaměstnán zbytečnými aktivitami, jako časté setkání s nadřízenými o diskutování možnosti plán dohnat, opakované přepracovávání odhadu k určení finálního data dokončení, omluvy za zpoždění, příprava demo vydání, opravy problémů vzniklých dřívější ukvapeností při vývoji, aj.

V praxi tedy potom vyvstává otázka, zda se uchýlit raději k nadhodnocení než k podhodnocení. Na obrázku níže je jasně vidět, že nejlepší výsledky jsou díky přesným odhadům (Obr. 2). V případě, že je odhad podhodnocen, neúčinnost plánování projektu zvýší skutečnou cenu a rozvrh projektu. Pokud je odhad naopak nadhodnocen, začne platit Parkinsonův zákon. Záměrné podhodnocení projektu kvůli snaze vyhnout se Parkinsonově zákonu má ale smysl pouze tehdy, je-li cena za nadhodnocení větší, než za jeho podhodnocení. V případě nadhodnocení je cena lineární a ohraničená, tedy práce se vyplňuje celý určený čas, ale dále již nepokračuje. Zatímco v případě podhodnocení je cena nelineární a neohraničená.



Obr. 2 – Podhodnocení vs. nadhodnocení projektu [1, str. 46]

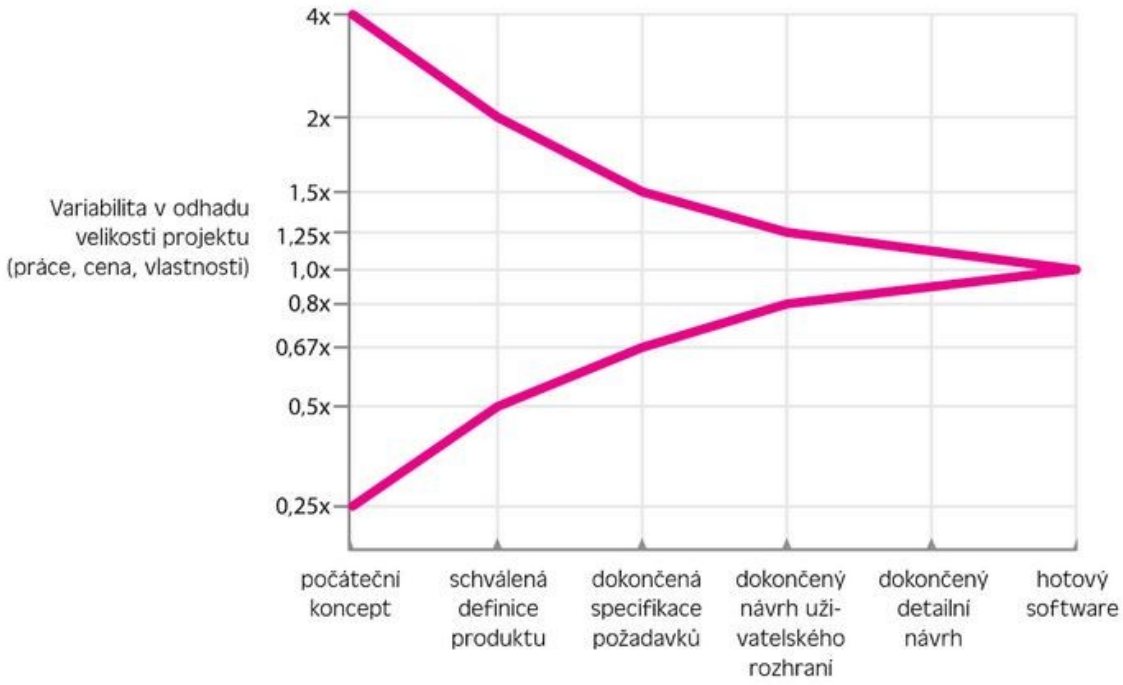
Pokud to situace umožňuje, je doporučeno záměrně nepodhodnocovat. Problémy u nadhodnocení je vhodno řešit raději správným plánováním a řízením vývoje projektu, nikoli posuny odhadů [1].

1.2.2 Chyby v odhadech

Chyby v odhadech vznikají díky nepřesným informacím o odhadovaném projektu, nepřesným informacím o možnostech vývojového oddělení, díky chaotickému projektu s pohyblivým cílem a díky nepřesnostem ze samotného procesu odhadování.

Aby bylo možné přesně odhadnout množství práce a tím stanovit cenu projektu, je třeba pochopit detailně vlastnosti softwarového projektu. Celý vývoj software je složen z postupného upřesňování. Na začátku je obecný koncept, jak by měl software vypadat a tento koncept se postupně formuje podle cílů projektu. Odhad je složen z postupných rozhodování o vlastnostech, které má splňovat. Vzniká zde nejistota o tom, jak daná rozhodnutí dopadnou.

Kužel nejistoty (Obr. 3) zobrazuje, jak se odhady v průběhu vývoje projektu zlepšují. Na horizontální jsou obvyklé milníky vývoje projektu od počátku až po hotový software, na vertikální ose je stupeň chyby, která je v odhadech v různých stupních vývoje.



Obr. 3 – Kužel nejistoty [1, str. 55]

1.2.3 Vlivy na odhad

Vlivy na softwarové projekty jsou oproti jiným projektům specifické [23] a jejich pochopení pomáhá docílit přesnějších odhadů. Lze je rozdělit několika způsoby [1]:

1. *Velikost projektu* – velikost vyvíjeného software má největší vliv na celý odhad. Velikost se může vyjadřovat v počtu řádcích kódu, v počtu funkcí, počtu požadavků,

počtu webových stránek, zkrátka v jakémkoli měřítku, který vyjádří stejný interval velikostí. V praxi tento vliv na celkovou cenu firmy často přehlížejí a provádějí odhady ceny, práce a rozvrhu bez znalosti velikosti daného software. Velikost projektu je třeba brát v potaz i v případě navýšení velikosti již probíhajícího projektu a cenu, práci a rozvrh upravit. Nevýhodou velikosti projektu je právě jeho velikost. Větší projekty vyžadují koordinaci větších skupin lidí, což má za následek kvadratický nárůst komunikačních kanálů. Mezi dvěma lidmi je jeden komunikační kanál, mezi třemi jsou 3 kanály, mezi čtyřmi lidmi je již 6 komunikačních kanálů, atd. Tento nárůst komunikačních kanálů vede rovněž ke kvadratickému nárůstu práce. V oblasti software platí, že čím je projekt větší, tím je jeho cena vyšší.

2. *Druh vyvíjeného software* – po velikosti vyvíjeného software druhý nejdůležitější aspekt při tvorbě odhadu. Je třeba při odhadu započítat, o jaký druh projektu jde, zda by jeho případné selhání bylo kritické pro životy nebo majetek. Na takový projekt bude zapotřebí započítat více úsilí, než například pro projekt bankovního systému, který bude mít stejnou velikosti.
3. *Lidský faktor* – rovněž lidský faktor je důležitým pro tvorbu odhadu a následných výsledků. Pokud ten, kdo je pověřen vyhotovením odhadu, nemá představu o tom, kdo bude na projektu pracovat, nemůže provést přesný odhad. Bylo vytvořeno mnoho studií (např. [7], [8], [9], [10]), které prokazují rozdíl ve výkonu jednotlivců a vývojových týmů v poměru 10:1: až 20:1.
4. *Programovací jazyk* – volba programovacího jazyka může odhad ovlivnit několika způsoby. Patří sem zkušenost vývojového týmu s daným jazykem, generování více funkcionality na jeden řádek oproti jiným jazykům nebo obsáhlost podpůrných nástrojů a prostředí daného jazyka.

2 METODY ODHADOVÁNÍ SOFTWAROVÝCH PROJEKTŮ

Při výběru vhodné metody odhadování je důležité zamyslet se nad tím, co vlastně chceme odhadovat. U některých projektů je potřeba odhadovat rozvrh a práci, u jiných projektů je k dispozici časový rámec a rozpočet a úkolem odhadu bude odhad množství funkčnosti, které lze dodat. Spousta metod odhadu lze aplikovat bez ohledu na to, co je vlastně cílem odhadu, a je také několik metod, které se lépe hodí pouze na odhad dané části projektu. Před výběrem metody je tedy nutné se prvně zamyslet nad několika otázkami:

1. Jak velký projekt budeme odhadovat?
2. Bude styl vývoje software postupný nebo opakující se?
3. V jaké fázi vývoje se projekt nachází?
4. Jak dosažitelnou přesnost od odhadu očekáváme?

2.1 Individuální expertní úsudek

V praxi patří individuální úsudek experta k nejčastějšímu přístupu pro tvorbu odhadů [1]. Odhady pro jednotlivé úkoly, vytvořené na základě úsudku experta, jsou základem odhadů typu zespodu nahoru. Je třeba však mít na paměti, že úsudek je nejriskantnější způsob pro odhadování. Tento úsudek, na jehož základě se tvoří odhad, vytváří expert. Zde se ale nabízí otázka, jaký expert? Na co je odborníkem? Fakt, že je někdo odborník na terminologii nebo způsob vývoje ještě neznamená, že je odborníkem taky na tvorbu odhadů v této oblasti. Magne Jørgensen uvádí [13], že větší zkušenosti v dané oblasti nevedou automaticky ke tvorbě lepších odhadů.

Tabulka 1 – Použitelnost metod individuálních expertních úsudků [1]

	Použití strukturova-ného procesu	**Použití souboru otázek o odhadu**	**Odhadování pracnosti úkolů v daném intervalu**	**Srovnávání odhadů úkolů a skutečnosti**
Co odhadujete	Práce, rozvrh, vlastnosti	Práce, rozvrh, vlastnosti	Velikost, práce, rozvrh, vlastnosti	Velikost, práce, rozvrh, vlastnosti
Velikost projektu	Malý, střední, velký	Malý, střední, velký	Malý, střední, velký	Malý, střední, velký
Fáze vývoje	Počáteční – pozdní	Počáteční – pozdní	Počáteční – pozdní	Střední – pozdní

Opakující se nebo postupný vývoj	Oba	Oba	Oba	Oba
Dosažitelná přesnost	Vysoká	Vysoká	Vysoká	Není k dispozici

Je asi logické, že nejlepšího odhadu dosáhne na základě úsudku ten, kdo bude následně činnost sám vykonávat. Je zde i menší šance podhodnocení daného úkolu [14]. Tohle pravidlo ale platí pouze v případě, že se odhadují jednotlivé úkoly. Pokud je projektu zatím v široké fázi kuželu nejistoty, a tedy zatím nejsou známy všechny jednotlivé úkoly ani nemají k sobě přiřazené odpovídající vývojáře, měl by odhad vytvořit buďto kvalifikovaným expertem na odhady, nebo nejlepším dostupným pracovníkem v oboru daného vývoje.

Nejlepší cestou ke zlepšení odhadů je rozklad na menší celky, kdy právě dojde k již zmiňovanému předejití podhodnocení odhadu. Je to dáno tím, že když testeři, vývojáři, zkrátka když příslušní lidé tvoří odhady v jim blízké oblasti, mají tendence soustředit se na úkoly, kterým rozumí a úkoly, které jim jsou cizí, neberou v potaz. Potom dochází k tomu, že místo plánovaných 2 týdnů zabere daný úkol například 2 měsíce. Když je ale projekt rozdělen na menší části, snáz se tomuto podhodnocení předejde. Ideální je rozložit si úkoly tak, aby nevyžadovaly více než 2 dny. Větší části by již mohly obsahovat hodně prostoru pro nečekanou práci, se kterou se v odhadu nepočítalo. Tyto jednotlivé úkoly poté vývojář musí ohodnotit číselnou hodnotou udávající, kolik času který úkol zabere. Kromě tohoto je také dobré požádat jej o vypracování druhého odhadu, který bude složen z intervalu možných dokončení. Z času, který úkol zabere v nejlepším případě a naopak času, kolik zabere v nejhorším případě. Jednočíselné odhady bývají většinou ve srovnání s těmito odhady pro nejlepší a nejhorší případ značně optimistické. Přemýšlení o tom, co vše by se mohlo v průběhu práce pokazit, někdy vede k rozeznání další práce, která je potřebná pro dokončení daného úkolu, a tím se zvyšuje původní jednočíselný odhad.

K ještě lepšímu odhadu se dá dojít určením nejen časového údaje pro nejlepší a nejhorší případ, ale určit také nejpravděpodobnější případ, v jakém je úkol možno splnit. Poté pomocí vzorce vypočítat hodnotu pro očekávaný případ (1):

$$Očekávaný = \frac{Nejlepší + (4 \times Nejpravděpodobnější) + Nejhorší}{6} \qquad (1)$$

Jelikož mají lidí tendenci své nejpravděpodobnější odhady podhodnocovat a vytvářet je optimistické, je navrženo upravení rovnice pro výpočet do tvaru (2):

$$Očekávaný = \frac{Nejlepší + (3 \times Nejpravděpodobnější) + (2 \times Nejhorší)}{6} \quad (2)$$

Upravená rovnice je však jen krátkodobým řešením. V dlouhodobějším řešení je doporučeno spolupracovat s takovými lidmi, kteří jsou schopni své nejpravděpodobnější odhady tvořit přesněji. K tomuto zpřesnění a zlepšování vytváření odhadů je vhodno si uchovávat seznam svých odhadů a po dokončení jej doplnit o skutečné výsledky. Pak lze jednoduše vypočítat velikost relativní chyby (MRE) daných odhadů:

$$MRE = \frac{SkutečnýVýsledek - OdhadovanýVýsledek}{SkutečnýVýsledek} \quad (3)$$

Na základě této chyby MRE lze sledovat, jak dobré odhady vývojář dělá a může se podle toho řídit v dalších odhadech [1]. Čím se budou odhady zlepšovat, tím by se měla hodnota MRE snižovat.

2.2 Skupinový expertní úsudek

Skupinové úsudky expertů jsou vhodné buďto v rané fázi projektu, nebo pro odhad velkých neznámých. Existují dvě metody, strukturované a nestrukturované metody.

Tabulka 2 – Použitelnost metod skupinových expertních úsudků [1]

	Skupinová hodnocení	**Wideband Delphi**
Co odhadujete	Velikost, práce, rozvrh, vlastnosti	Velikost, práce, rozvrh, vlastnosti
Velikost projektu	Střední, velký	Střední, velký
Fáze vývoje	Počáteční – střední	Počáteční
Opakující se nebo postupný vývoj	Oba	Postupný
Dosažitelná přesnost	Střední	Střední

2.2.1 Skupinová hodnocení

Metoda tvorby odhadu, kdy se skupinově hodnotí nestrukturované individuální úsudky expertů. Při hodnocení odhadů je třeba dodržet tato tři pravidla [1]:

1. Každý člen týmu provede individuální odhad jednotlivých částí projektu. U následného porovnávání odhadů je kladen důraz na prodiskutování všech rozdílů v odhadech s cílem odhalení zdroje těchto rozdílů.
2. Prodiskutování rozdílů v jednotlivých odhadech, namísto pouhého spokojení se zprůměrováním odhadů.
3. Dojití k souhlasu s odhadem od všech členů týmu. V případě nejasností je třeba rozebrat rozdíly a získat souhlas od každého člena.

Na zlepšení individuálních odhadů je tedy určitě dobré použít skupinové hodnocení, přičemž skupina by se měla skládat minimálně z 3 až 5 expertů.

2.2.2 Wideband Delphi

Metoda Wideband Delphi je vhodná pro odhady v rané fázi projektu. Jedná se o strukturovanou metodu odhadování, vyvinutou pro předpovídání trendů v technologii na konci čtyřicátých let 20. století ve společnosti Rand Corporation [15]. Základní metoda svolává několik odborníků, kteří mají za úkol vytvořit nezávislé odhady a následně o nich diskutovat, dokud se jejich odhady buďto nesetkají nebo dokud se nedomluví na stejném výsledku. Základní verze je tedy vesměs stejná, jako výše zmíněné skupinové hodnocení. Boehm a jeho spolupracovníci proto tuto verzi rozšířili, a tak vznikla verze Wideband Delphi.

Hlavní procedura techniky [1]:

1. Každému odhadovateli je koordinátorem Delphi předložena specifikace a forma odhadu.
2. Odhadovatelé připraví své individuální odhady. Tento krok lze případně provést až po kroku 3.
3. Koordinátorem je svoláno skupinové setkání, kde spolu odhadovatelé probírají problémy, které úzce souvisí s projektem. Pokud se skupina shodne na jednoduchém odhadu bez velkých diskuzí, koordinátor někomu přiřadí roli ďáblova advokáta.
4. Odhadovatelé dají koordinátorovi anonymně své individuální odhady.
5. Koordinátor připraví souhrn odhadů a předloží jej odhadovatelům.
6. Koordinátor nechá prostor, aby mohli odhadovatelé rozebrat odchylky ve svých odhadech.
7. Odhadovatelé anonymně hlasují, zda přijmout průměrný odhad. V případě, že někdo hlasuje pro „ne“, proces se vrací do bodu 3.

8. Finální odhad je jednočíselný a pramení z této procedury Delphi. Nebo je finální odhad interval vytvořený v průběhu diskuze Delphi a jednočíselný odhad Delphi je očekávaný případ.

Kroky 3 až 7 mohou být provedeny osobně, na skupinových setkáních, nebo pro větší zachování anonymity elektronicky e-mailem či v diskuzní skupině. V závislosti na tom, jak rychle je potřeba odhad vytvořit, mohou být tyto kroky provedeny ihned po sobě nebo s jistým odstupem.

Tato metoda odhadu se hodí k odhadu práce v nové obchodní oblasti, při práci s novou technologií, při vývoji nového druhu software. Jak už bylo řečeno, je vhodná pro rané fáze projektu, kdy ještě nejsou mnohé požadavky stanoveny. Vhodná je rovněž v případě, že bude projekt obsahovat mnoho neobvyklých řešení, jako například kombinace nestandardní použitelnosti, algoritmické složitosti, výjimečného výkonu, složitých obchodních podmínek aj.

Nevýhoda této metody je její nákladnost na čas pracovníků, jelikož vyžaduje vzájemné diskuze. Není vhodná pro detailní odhady úkolů.

2.3 Algoritmické metody

Algoritmické metody využívají matematický vzorec pro odhad nákladů projektu na základě odhadu velikosti projektu, počtu softwarových inženýrů a dalších procesních a výrobních faktorů. Algoritmický nákladový model lze sestavit analýzou historických dat. Algoritmické metody se primárně používají k odhadům nákladů na vývoj software. Jejich použití je vhodné u středních a větších projektů.

Do algoritmických metod patří například COCOMO 81, COCOMO II, Use Case Points, funkční bodová analýza FPA, Putnamův model nebo Bailey-Basilievův model.

2.3.1 COCOMO II

Konstruktivní nákladový model COCOMO (COnstructive COst MOdel) vytvořil v roce 1981 Barry Boehm a ve své době byl jedním z nejpopulárnějších parametrických modelů pro odhady nákladů na vývoj softwarových projektů [21]. Původní model v roce 1995 nahradil model COCOMO II. Tento model byl již vytvořen tak, aby byl evoluční, na rozdíl od původního modelu, kterému se pro přehlednost říká COCOMO 81. Nová metoda modelu tedy již může být aktualizována a upravována dle aktuálních změn při vývoji software.

COCOMO II se skládá ze tří dílčích modelů [4]:

1. **Early Design model (EDM)** – tento model je vhodné použít v raných fázích projektu, kdy se architektura celého projektu teprve plánuje. V této fázi se toho o povaze daného projektu i o softwarových produktech, které mají být dodány, ví velmi málo. Jde o takové pojmy, jako je velikost softwarových produktů, cílový program, sestava vývojového týmu, který se bude na projektu podílet nebo třeba podrobné specifikace procesů, které mají být použity. Jelikož se pracuje pouze s malým množstvím informací, vyžaduje model EDM menší množství dat než model PAM, díky čemuž ale poskytuje méně přesné odhady.
2. **Post-Architecture model (PAM)** – tento model představuje podrobný odhad modelu a je určen pro pozdější fáze projektu, kdy již byla vytvořena celková softwarová struktura a kdy je tedy software připraven k vývoji. K vypracování podrobnějších a především přesnějších odhadů vyžaduje model více vstupních informací.
3. **Application Point model (APM)** – tento model je rozšiřující a je potřeba v případě použití moderních nástrojů pro rychlý vývoj software, jako jsou RAD (Rapid Application Development) nebo GUI (Graphic User Interface).

Základní model COCOMO II je reprezentován statistickým regresním modelem (4):

$$Effort = A \times Size^{E} \times \prod_{i=1}^{n} EM_i \tag{4}$$

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j \tag{5}$$

kde

Effort	Celkové plánované úsilí
A	Produktivita vývoje, počáteční hodnota A = 2,94
Size	Velikost softwarového produktu
E	Efekt rozsahu
EM	Násobiče úsilí
B	Pomocná konstanta, počáteční hodnota B = 0,91
SF	Faktory velikosti

Celkové plánované úsilí Effort (4) se měří v člověkoměsících. K jeho výpočtu je zapotřebí znalost dvou pomocných konstant A a B, velikost software udávaná v tisících řádků kódu

(KLOC) a znalost parametrů EM a SF. COCOMO II nabízí i jiné možnosti pro změnu velikosti, například funkční nebo objektové body, ale i ty jsou před provedením odhadu vnitřně převedeny na KLOC. Konstanty A a B jsou pro tuto verzi modelu nastaveny na počátečních hodnotách 2,94 a 0,91. Konstanty lze upravit pro konkrétní situaci pomocí historických dat, což je ostatně i doporučeno k dosažení nejjednoduššího způsobu kalibrace modelu pro konkrétní organizaci.

Model COCOMO II uvažuje 5 faktorů velikosti (SF), které mají dopad na exponenciální kolísání produktivity software a úsilí [4]:

1. **PREC** – Precedentedness – rozsah podobnosti s již dokončenými projekty. Faktor zahrnuje aspekty jako organizační chápání cílů produktů, zkušenosti s prací se softwarovými systémy, souběžný vývoj nového hardware a software, inovativní zpracování algoritmů, aj.
2. **FLEX** – Development flexibility – flexibilita vývoje udává do jaké míry je softwarový produkt osvobozen od potřeby být v souladu se zavedenými požadavky.
3. **RESL** – Architecture/risk resolution – faktor představující vážený průměr několika charakteristik projektu, věnuje se řešení architektury projektu a jeho rizik. Například do jaké míry plán řízení rizik identifikuje všechny kritické projektové rizika, do jaké míry plán řízení stanoví plán, rozpočet a vnitřní milníky pro řešení, rozsah podpory nástrojů pro řešení rizik, úroveň nejistoty klíčových faktorů architektury, aj.
4. **TEAM** – Team cohesion – faktor udávající úroveň obtížnosti v synchronizaci zainteresované strany projektu (uživatelé, zákazníci, vývojáři, správci, projektanti, aj.). Problémy vznikají, když mají tyto strany odlišné vize a cíle, omezené schopnosti nebo neochotu vyhovět cílům ostatních stran, popřípadě když chybí týmové zkušenosti a znalosti.
5. **PMAT** - Process maturity – faktor určující zralost procesů vývoje software z hlediska modelu zralosti způsobilosti (CMM). Model má 5 stupňů, a jeho cílem je zdokonalení procesů vývoje software.

Pro každý faktor se určí jeho vliv na stupnici od velmi nízký, až po extrémně vysoký. Každý stupeň ohodnocení je spojen s konkrétní hodnotou, která je přičtena k celkové hodnotě E. V závislosti na výsledku E (5) může dojít ke třem možnostem:

1. $E < 0$, projekt vykazuje úspory v rozsahu
2. $E = 0$, dosažení rovnováhy mezi velikostí software a úsilím projektu

3. E > 0, projekt vykazuje záporné úspory v rozsahu

Násobiče úsilí (EM) představují vlastnosti vývojového prostředí, které má významný dopad na základní produktivitu vývoje a tím pádem i na celkové úsilí. Počet těchto násobičů je jediné, v čem se modely EDM a PAM liší. ED model má celkem 7 násobičů úsilí, PA model jich má 17. Podobně jako faktory velikosti, mají i násobiče úsilí přiřazené kvantitativní hodnocení úrovní, např. nízká, nominální nebo vysoká. Každý stupeň je opět ohodnocen konkrétní hodnotou, tzv. násobičem úsilí. Například nominální hodnocení znamená, že faktor nemá ani kladný ani záporný dopad na úsilí, a proto je spojen s hodnotou 1. Pokud má faktor kladný dopad na úsilí, je ohodnocen vyšší hodnotou, než je 1. Pokud má záporný dopad, tak je odpovídající hodnota nižší než 1.

Násobiče úsilí pro PAM jsou rozděleny do čtyř skupin [4]:

1. Faktory produktu
   - **RELY** - Required software reliability – požadovaná spolehlivost fungování softwaru, hodnocení v rozsahu od 0,82 do 1,26
   - **DATA** - Database size – čím větší velikost databáze, tím vyžaduje více práce. Hodnocení v rozsahu 0,90 až 1,28.
   - **CPLX** - Product complexity – složitost produktu je nejdůležitějším nastavitelným prvkem a je z velké části dána typem vyvíjeného software. Rozsah hodnocení 0,73 až 1,74.
   - **RUSE** - Developed for reusability – software, který je vyvíjen pro opakované použití, může být až o třetinu dražší. Rozsah hodnocení 0,95 až 1,24.
   - **DOCU** - Documentation match to life cycle needs – rozsah dokumentace pro jednotlivé životní cykly softwaru. Příliš obsáhlá dokumentace může negativně ovlivnit celý projekt. Rozsah hodnocení 0,81 až 1,23.
2. Faktory platformy
   - **TIME** - Execution time constraint – omezená doba odezvy, minimalizací zvýšíme celkovou práci. Rozsah hodnocení 1,00 až 1,63.
   - **STOR** - Main storage constraint – omezená kapacita uložiště platformy mírně zvyšuje množství potřebné práce. Rozsah hodnocení 1,00 až 1,46.
   - **PVOL** - Platform volatility – v případě nestabilní platformy může dojít k prodloužení vývoje. Rozsah hodnocení 0,87 až 1,30.

3. Charakteristika lidí
   - **ACAP** – Analyst capability – schopnosti analytiků jsou hodnoceny v rozsahu od 1,42 (velmi nízké) až do 0,71 (velmi vysoké) v závislosti na schopnostech analyzovat, navrhovat řešení, spolupracovat a komunikovat s ostatními kolegy.
   - **PCAP** – Programmer capability – schopnosti programátorů jsou hodnoceny v rozsahu 1,34 až 0,76.
   - **PCON** – Personnel continuity – soudržnost týmu je hodnocena v rozsahu 1,29 až 0,81
   - **AEXP** – Applications experience – zkušenost s aplikacemi je hodnocena v rozsahu 1,22 až 0,81. Čím méně zkušeností v dané oblasti, tím více času projekt zabere.
   - **PEXP** – Platform experience – zkušenosti s platformou, na které se provádí vývoj, jsou hodnoceny v rozsahu 1,19 až 0,85.
   - **LTEX** – Language and tool experience – zkušenosti s jazykem a nástroji jsou hodnoceny od 1,20 do 0,84.
4. Faktory projektu
   - **TOOL** – Use of software tools – použití softwarových nástrojů ovlivňuje úsilí v rozsahu 1,17 až 0,78.
   - **SITE** – Multisite development – projekty vyvíjeny na více místech potřebují až o 56% více práce než projekty vyvíjeny na jednom místě. Rozsah hodnocení 1,22 až 0,78.
   - **SCED** – Required development schedule – požadovaný termín dokončení je hodnocen v rozsahu od 1,43 do 1,00. čím kratší doba projektu, tím vyšší úsilí.

Násobičů úsilí pro EDM je 7 a získají se kombinací výše zmíněných faktorů pro PAM [4]:

1. **PERS** – Personnel capability – schopnosti personálu jsou tvořeny kombinací ACAP, PCAP a PCON
2. **RCPX** - Product reliability and complexity – složitosti a spolehlivost software je kombinací faktorů RELY, DATA, CPLX a DOCU
3. **PDIF** - Platform difficulty – složitost platformy je tvořena kombinací TIME, STOR a PVOL
4. **PREX** – Personnel experience – zkušenosti personálu jsou kombinací AEXP, PEXP a LTEX
5. **FCIL** – Facilities – vybavenost je kombinací TOOLS a SITE

6. **RUSE** - Developed for reusability – vývoj pro opakované použití, stejné jako v PA modelu
7. **SCED** - Required development schedule – požadovaný termín dokončení je rovněž stejný jako v PA modelu

Užití základního tvaru modelu COCOMO II v rámci určité organizace může vést k nepřesným odhadům. Pro zmírnění tohoto rizika by měl být model kalibrován pomocí historických dat dané organizace, ve které má fungovat. Nutnost kalibrace se vztahuje zejména na PA model, který je určen k poskytování přesných odhadů. Barry Boehm uvádí tři základní způsoby pro kalibraci PA modelu:

1. Kalibrace multiplikativní konstanty pomocí historických dat
2. Sloučení nebo odstranění nadbytečných nebo nepodstatných parametrů modelu
3. Přidání relevantních parametrů modelu, které ještě nebyly modelovány

Kalibrace pomocí historických dat zahrnuje úpravu konstanta A a B. Je zapotřebí mít k dispozici minimálně 5 starších projektů pro kalibraci konstanty A. V případě kalibrace obou konstant je zapotřebí minimálně 10 projektů.

Ke sloučení parametrů dojde tehdy, jsou-li pokládány za jeden. Například určitá organizace nemusí rozlišovat mezi faktory omezení odezvy (TIME) a omezení kapacity uložiště (STOR), a tak tyto dva faktory sloučí do jednoho nového faktoru. K odstranění dochází tehdy, je-li faktor ve všech projektech organizace ohodnocen stejně. V tomto případě faktor nezpůsobuje žádnou odchylku ve vývoji produktivity v rámci více projektů a může být tedy z modelu vyjmut.

K přidání nového multiplikátoru dojde tehdy, když v obecném COCOMO II chybí nějaký důležitý faktor, například úroveň zabezpečení software. Pro nový faktor musí být stanovena vhodná stupnice hodnocení a nový faktor musí být kalibrován pomocí historických dat nebo expertního posudku.

2.3.2 FPA

Analýza funkčních bodů byla vyvinuta poprvé v polovině sedmdesátých let Allanem J. Albrechtem a publikována byla poprvé v roce 1979 [20]. Na rozdíl od metody COCOMO nepotřebuje k odhadu znát počet řádků kódu, ale bere do úvahy vstupní a výstupní prvky projektu a zaměřuje se na funkční požadavky systému. Metoda ověřuje jednotlivé prvky a související skupiny tak, aby dosáhly tří úrovní složitostí (vysoké, průměrné, nízké) a přiřazuje

počítání funkčních bodů pro každou podmnožinu. Celkový součet vysokého, průměrného a nízkého počtu všech operací dává výpočet neupravených funkčních bodů.

Unadjusted Function Points (UFP) – Neupravené funkční body

Existuje pět základních typů funkcí, které jsou rozděleny do dvou hlavních kategorií [19]:

1. Datové funkce – funkce týkající se ukládání a načítání dat v místních souborech nebo databázích a externích aplikací prostřednictvím vzdálených rozhraní, přidružených středních produktů nebo mimo hranice příslušné aplikace
   - **Internal Logical Files (ILF)** – jedná se o interní logický soubor, uživatelsky identifikovatelnou skupinu logicky příbuzných dat, která se nachází zcela v rámci hranice vyvíjené aplikace. Data jsou zde uchována prostřednictvím elementárního procesu, který tvoří možnosti vytvořit, číst, editovat a smazat operaci v rámci hranic aplikace.
   - **External Interface Files (EIF)** – soubor externího rozhraní je uživatelsky identifikovatelná skupina logicky souvisejících dat nebo řídících informací, na které odkazuje aplikace, ale udržuje se v hranici jiné aplikace.
2. Transakční funkce – transakční funkce jsou funkce týkající se operací čtení a zápisu prováděných na datech, probíhajících mezi datovými funkcemi ILF nebo EIF.
   - **External Input (EI)** – základní proces zpracovávající data nebo řídící informace, které jsou mimo hranice aplikace. Data mohou pocházet z obrazovky pro zadávání dat nebo z jiné aplikace.
   - **External Output (EO)** – základní proces, ve kterém data prochází přes hranici projektu ven. Jedná se o operace čtení dat a výsledek může být vytištěn, přenášen na externím zařízení nebo v jiné aplikaci.
   - **External Inquiry (EQ)** – externí dotaz je proces, který se věnuje odesílání dat nebo řídících informací mimo hranice projektu. Na rozdíl od externích výstupů, externí dotazy nemohou obsahovat nějakou matematickou rovnici, výpočet součtu, průměr, počet nebo jinou manipulaci s daty, nebo vytvářet další odvozená pole (součty, mezisoučty, výpočet konečných nákladů) nebo aktualizovat ILF tak, aby odrážely vypočítanou částku.

Při výpočtu neupravených funkčních bodů je potřeba nejprve nalézt všechny používané funkce a rozdělit je do jednotlivých skupin podle jejich složitosti. Po tomto dosazení je vynásoben počet prvků příslušnou váhou a sečtením všech těchto hodnot se získá neupravený počet bodů (Tabulka 3).

Tabulka 3 – Výpočet neupravených funkčních bodů [20]

Typ komponenty	**Složitost komponenty**			**Součet**
	Nízká	Průměrná	Vysoká	
EI	_ x 3 = _	_ x 4 = _	_ x 6 = _	
EO	_ x 4 = _	_ x 5 = _	_ x 7 = _	
EQ	_ x 3 = _	_ x 4 = _	_ x 6 = _	
ILF	_ x 7 = _	_ x 10 = _	_ x 15 = _	
EIF	_ x 5 = _	_ x 7 = _	_ x 10 = _	
Celkový součet neupravených funkčních bodů (UFP)				

Adjusted Function Points (AFP) – Upravené funkční body

Faktor složitosti může ovlivnit konečný výsledek o ± 35%. Za konečný výsledek je považován výpočet upravených funkčních bodů. Používá se celkem 14 obecných systémových charakteristik k identifikaci složitosti aplikace (Tabulka 4), jejichž vliv je ohodnocen na stupnici od 0 do 5.

Tabulka 4 – Seznam faktorů pro AFP

Typ	**Vliv (Ci)**	**Typ**	**Vliv (Ci)**
Datová komunikace	<0,5>	Přímá oprava	<0,5>
Distribuované zpracování dat	<0,5>	Složitost zpracování	<0,5>
Výkonnost	<0,5>	Opakovaně použitelný kód	<0,5>
Plně využitá konfigurace	<0,5>	Snadná instalace	<0,5>
Rychlost zpracování transakcí	<0,5>	Snadné použití	<0,5>
Přímé vstupy dat	<0,5>	Možnost nasazení	<0,5>
Efektivita koncového uživatele	<0,5>	Možnost změny	<0,5>

Hodnota upravených funkčních bodů se vypočítá pomocí vzorce (6), kde se faktor nastavení

hodnoty VAF (Value Adjusted Factor) vypočítá pomocí vzorce (7) a celkový součet vlivů faktorů TDI (Total Degree of Influence) se vypočítá podle rovnice (8):

$$AFP = UFP \times VAF \tag{6}$$

$$VAF = 0{,}65 + (TDI \times 0{,}01) \tag{7}$$

$$TDI = \sum_{i=1}^{14} Ci \tag{8}$$

Odhad doby vývoje projektu se vypočítá pomocí vzorce (9):

$$T = AFP^{0,4}\ [měsíce] \tag{9}$$

2.3.3 UCP

Metodu Use Case Points vyvinul v roce 1993 Gustav Karner [16]. Metoda je založena na stejné filozofii jako analýza funkčních bodů a používá se pro odhadování projektů vyvinutých pomocí objektově orientované metody.

Odhad metodou UCP se vypočítá podle vzorce (10):

$$UCP = (UAW + UUCW) \times TCF \times ECF \tag{10}$$

kde

UAW	Celková neupravená váha aktérů
UUCW	Celková neupravená váha případů užití
TCF	Technický faktor
ECF	Faktor prostředí

Na začátku metody jsou aktéři a případy užití rozděleni do třech kategorií podle své složitosti, na základě které je jim následně přiřazena váha [17]. Jednoduchý aktér může představovat aplikační programové rozhraní (API), systém propojený síťovým protokolem představuje průměrného aktéra a komplexní aktér představuje osobu komunikující přes uživatelské rozhraní nebo webové stránky. Klasifikace aktérů a přiřazení vah je shrnuto v tabulce níže (Tabulka 5).

Celková neupravená váha aktérů UAW se vypočítá pomocí vzorce (11):

$$UAW = \sum AC \times WFa \tag{11}$$

Tabulka 5 – Klasifikace aktéra a jeho váha

Klasifikace aktéra (AC)	**Váha (WFa)**
Jednoduchý	1
Průměrný	2
Komplexní	3

Případy užití jsou zpracovány stejným způsobem (Tabulka 6). Složitost případu užití je závislá na počtu kroků neboli transakcí [28]. Původní metoda počítá kroky v hlavním i alternativním scénáři. Pokud je případ užití rozšířen, nebo je součástí jiného případu užití, tyto kroky nejsou započteny a počítají se jako samostatné scénáře.

Celková neupravená váha případů užití UUCW se vypočítá pomocí vzorce (12):

$$UUCW = \sum UCC \times WFb \tag{12}$$

Tabulka 6 – Klasifikace případu užití a jeho váha

Klasifikace případu užití (UCC)	**Počet kroků**	**Váha (WFb)**
Jednoduchý	(0,4)	5
Průměrný	<4,7>	10
Komplexní	(7,∞)	15

Následují dvě korekční hodnoty – technický faktor a faktor prostředí [22], které je potřeba rovněž vypočítat. Pro oba tyto faktory může odhadovatel individuálně definovat rozsah dopadu na projekt od 0 do 5, kde 0 znamená zanedbatelný dopad, 3 průměrný dopad a 5 velmi významný dopad.

Technický faktor TCF se vypočítá pomocí vzorce (13):

$$TCF = 0{,}6 + (0{,}01 \times \sum_{T13}^{T1} WFc \times SIa) \tag{13}$$

Technických faktorů je celkem 13 a jejich přehled je uveden níže (Tabulka 7).

Faktory prostředí ECF mají, stejně jako technické faktory, pevně definovanou hodnotu vah (Tabulka 8). Určení dopadu každého faktoru je určeno stejným způsobem, jako u TCF.

Tabulka 7 – Technické faktory

Faktor	Popis	Váha (WFc)	Dopad (SIa)
T1	Distribuovaný systém	2	<0,5>
T2	Odpověď adjektiv	2	<0,5>
T3	Efektivita koncového uživatele	1	<0,5>
T4	Komplexní zpracování	1	<0,5>
T5	Opakovaně použitelný kód	1	<0,5>
T6	Snadná instalace	0,5	<0,5>
T7	Snadné použití	0,5	<0,5>
T8	Přenosný / portable	2	<0,5>
T9	Snadná změna	1	<0,5>
T10	konkurenceschopný	1	<0,5>
T11	Bezpečnostní funkce	1	<0,5>
T12	Dostupnost třetím stranám	1	<0,5>
T13	Nutnost speciálních školení	1	<0,5>

Tabulka 8 – Faktory prostředí

Faktor	Popis	Váha (WFc)	Dopad (SIa)
E1	Obeznámení s RUP	1,5	<0,5>
E2	Zkušenosti s aplikací	0,5	<0,5>
E3	Objektově orientované schopnosti	1	<0,5>
E4	Schopnosti vedoucího analytika	0,5	<0,5>
E5	Motivace	1	<0,5>
E6	Stabilní požadavky	2	<0,5>
E7	Zaměstnanci na částečný úvazek	-1	<0,5>
E8	Složitost programovacího jazyka	2	<0,5>

Faktor prostředí ECF se poté určí pomocí vzorce (14):

$$ECF = 1{,}4 + (-0{,}03 \times \sum_{E8}^{E1} WFd \times SIb) \qquad (14)$$

Výsledná hodnota UCP udává velikost projektu. Kromě velikosti je ovšem potřeba zjistit také dobu trvání projektu, a ta z výsledku jasná není. Z tohoto důvodu je potřeba výsledek ještě vynásobit faktorem produktivity PF. Karner tento faktor navrhnul na 20 hodin na jeden bod případu užití. Kirsten Ribu [18] uvádí rozsah pro tento faktor od 15 do 30 hodin, což do převodu na hodiny lidské práce přináší určitou nejistotu. K vhodnému určení faktoru je doporučeno, namísto pouhého odhadování hodin na jeden případ užití, vypočítat průměrnou hodnotu pomocí historických dat dané organizace [17].

2.4 Kalibrace a historická data

Odhady vždy obsahují určitý druh kalibrace, která se používá pro převod počtů na odhady, ať už počet řádků kódu na práci, počet uživatelských celků na kalendářní čas, atd. Odhady lze kalibrovat třemi druhy dat, jak je zobrazeno v přehledové tabulce níže.

Tabulka 9 – Použitelnost metod na základě kalibrace a historických dat [1]

	Kalibrace pomocí průměrných dat z celého odvětví	**Kalibrace pomocí firemních dat**	**Kalibrace pomocí dat z konkrétního projektu**
Co odhadujete	Velikost, práce, rozvrh, vlastnosti	Velikost, práce, rozvrh, vlastnosti	Velikost, práce, rozvrh, vlastnosti
Velikost projektu	Malý, střední, velký	Malý, střední, velký	Malý, střední, velký
Fáze vývoje	Počáteční - střední	Počáteční - střední	Střední – pozdní
Opakující se nebo postupný vývoj	Oba	Oba	Oba
Dosažitelná přesnost	Nízká – střední	Střední – vysoká	vysoká

Kalibrace pomocí průměrných dat z celého odvětví je dočasná alternativa v případech, kdy nejsou k dispozici žádná jiná data [1]. Velmi užitečná pro tvorbu velmi přesných odhadů je kalibrace pomocí firemních dat, neboli kalibrace pomocí historických dat, a kalibrace pomocí dat z konkrétního projektu.

Používání historických dat při tvorbě odhadu výrazně zvyšuje jeho přesnost, a to hned z několika důvodů. Jedná-li se o malý projekt, jeho dokončení je silně závislé na individuálních schopnostech vývojáře. S velikostí projektu roste, kromě individuálních schopností jedinců,

také vliv charakteristiky dané organizace. Firemní vliv ovlivňující dokončení projektu je například [1]:

1. Schopnost firmy zajištění stability požadavků
2. Pravomoc projektového manažera odvolat z projektu problémového pracovníka
3. Zajištění dostatečného množství pracovníků pro projekt
4. Podpora efektivních postupů
5. Spoleh na to, zda všichni členové týmu zůstanou až do dokončení projektu ve firmě

Historická data započítání všech firemních vlivů usnadňují, jelikož tyto vlivy už obsahují. Pomáhají také předcházet subjektivitě a nepodloženému optimismu. Subjektivita se do odhadů vkrádá například po srovnání nového projektu se starým, kdy dojdou manažeři k tomu, že je mezi nimi mnoho rozdílů a na tomto srovnání dojdou k závěru, že nový projekt půjde udělat lépe. Vytvoří tak odhad, ve kterém nebudou počítat s možnými problémy, s odchodem zaměstnanců před dokončením projektu, se spoustou požadavků dodaných na poslední chvíli a podobně. Použitím historických dat se dá celý proces zjednodušit. Stačí vycházet z předpokladu, že nový projekt poběží zhruba stejně jako projekty předchozí.

Dalším problémem při tvorbě odhadu je fakt, že spousta nastavovacích prvků, které mají velkou váhu, závisí na lidech. Například model COCOMO II vyžaduje ohodnocení schopností všech zaměstnanců, kteří se budou na projektu účastnit. Toto ohodnocení je tedy zcela závislé na tom, jak manažer tým pracovníků vnímá a na kolik je objektivní. Použitím historických dat se však lze případným neshodám kvůli ohodnocení daného týmu vyhnout.

Historická data jsou tedy užitečnou pomůckou pro tvorbu velmi přesných odhadů, jelikož obsahují vlivy dané organizace, jak zjevné, tak i skryté. Stejným způsobem lze aplikovat historická data z daného projektu přímo na ten stejný projekt, díky čemuž bude mít projekt trochu odlišnou dynamiku od jiných projektů. Historická data z daného projektu obsahují i specifické vlivy pro daný projekt, a čím dřív se začnou odhady stavět na vlastních historických datech, tím dříve se docílí skutečně přesných odhadů.

2.5 Odhady pomocí zástupce

Metody odhadování pomocí zástupce pomáhají překonat problémy při odhadování, jako je například problém při pouhém podívání se na popis vlastnosti přesně odhadnout, kolik bude tato vlastnost vyžadovat řádků kódu. Při odhadech pomocí zástupce je potřeba nejprve nalézt vhodného zástupce, který odpovídá tomu, co se odhaduje. Pomocí něj pak lze na základě

historických dat dopočítat to, co nás v odhadu projektu zajímá. V tabulce níže je přehled nejužitnějších metod, které budou postupně v této kapitole popsány.

Tabulka 10 – Použitelnost metod odhadů pomocí zástupce [1]

	Fuzzy logika	**Standardní komponenty**	**Story points**	**T-shirt sizing**
Co odhadujete	Velikost, vlastnosti	Velikost, práce	Velikost, práce, rozvrh, vlastnosti	Práce, cena, rozvrh, vlastnosti
Velikost projektu	Střední, velký	Malý, střední, velký	Malý, střední, velký	Střední, velký
Fáze vývoje	Počáteční	Počáteční – střední	Počáteční – střední	Počáteční
Opakující se nebo postupný vývoj	Postupný	Oba	Oba	Postupný
Dosažitelná přesnost	Střední	Střední	Střední – vysoká	Není k dispozici

2.5.1 Fuzzy logika

Odhad pomocí fuzzy logiky se používá pro odhad celého projektu, nikoli pouze některých jeho částí. Aby tento odhad fungoval dobře, je zapotřebí použít jej na projekt s alespoň 20 vlastnostmi k odhadování. V případě, že je vlastností méně, statistika v této metodě nepracuje správně. Fuzzy logika lze použít například pro odhad velikosti projektu v řádcích kódu [11], [12].

K vytvoření takového odhadu je možné použít historická data z dané společnosti. Na začátku odhadu se každá požadovaná vlastnost ohodnotí od velmi malé až po velmi velkou. Z historických dat si z předchozích projektů vedoucí odhadu zjistí, jaký je průměrný počet řádků kódu pro jednotlivé velikosti vlastností. K docílení co největší důslednosti je ideální, aby byla pro tyto velikosti vytvořena specifická kritéria.

Tabulka 11 – Příklad použití fuzzy logiky pro odhad velikosti programu [1]

Velikost vlastnosti	**Průměrný počet řádků kódu na vlastnost**	**Počet vlastností**	**Odhad počtu řádků kódu**
Velmi malá	127	22	2 794

Malá	253	15	3 795
Střední	500	10	5 000
Velká	1 014	30	30 420
Velmi velká	1 998	27	53 946
Součet	**-**	**104**	**95 955**

V prvním sloupci jsou definovány jednotlivé velikosti vlastností. Druhý sloupec obsahuje průměrný počet řádků kódu na danou velikost vlastnosti. Tato hodnota je dopočítána z historických dat následujícím způsobem. Ve všech minulých projektech jsou vlastnosti ohodnoceny podle jejich velikosti. Následně je proveden součet všech vlastností v dané kategorii velikosti. Poté je pro každou velikost zjištěn celkový počet řádků kódu a na závěr je počet řádků vydělen počtem vlastností v každé kategorii. Tímto postupem se dostaneme k průměrnému počtu řádku každé kategorie velikosti. Poměry mezi velikostmi dvou sousedících kategorií by měly být alespoň 2, některými odborníky je doporučován dokonce poměr rozdílnosti alespoň 4 [11]. Ve třetím sloupci je definován počet vlastnosti v daných kategoriích v projektu, který chceme odhadovat. Po vynásobení průměrného počtu řádků na jednu vlastnost s počtem vlastností se získá poslední sloupec s odhadem počtu řádků. Součet je v tomto ukázkovém případě 95 955 řádků kódu. Výsledná velikost tohoto odhadu by, po zaokrouhlení k předejití pocitu přesnosti, byla 96 000 řádků kódu, nebo klidně i 100 000 řádků kódu.

Na základě historických dat lze tato metodika tvorby odhadu aplikovat také například pro odhady práce [1].

2.5.2 Standardní komponenty

Metoda odhadu pomocí standardních komponent je vhodná, pokud se ve společnosti vyvíjí projekty, které jsou si architektonicky podobné. Její aplikace je vhodná díky její jednoduchosti zejména v brzkých fázích projektu, kdy odhady, jak ukazuje kužel nejistoty, stejně podléhají vysokému stupni nepřesnosti.

Základem je z předchozích projektů tyto standardní komponenty určit, což je čistě individuální proces každé společnosti. Typický systém by mohl obsahovat dynamické a statické webové stránky, soubory, databázové tabulky, obchodní pravidla, grafiku, reporty, atd. Po určení těchto standardních komponent je v předchozích projektech spočítán průměrný počet řádků kódu na komponentu. Na základě těchto historických dat se dá odhadnout počet standardních komponent v novém projektu a dopočítat celková velikost projektu. Do tabulky pro

výpočet velikosti projektu je potřeba vložit odhadované počty jednotlivých komponent – nejprve minimální počet, který si je možno v projektu představit, poté nejpravděpodobnější počet a na konec maximální představitelný počet. Z těchto hodnot se vypočítá odhadovaný počet komponent pomocí vzorce (15):

$$K = \frac{[K_{min} + (4xK_{best}) + K_{max}]}{6} \tag{15}$$

kde

K	Odhadovaný počet komponent
K_{min}	Minimální možný počet komponent
K_{best}	Nejpravděpodobnější počet komponent
K_{max}	Maximální možný počet komponent

Velikost se pak už dopočítá prostým vynásobením počtu odhadnutých komponent s počtem řádků na jednu komponentu [1].

2.5.3 Story points

Story points je jiná varianta fuzzy logiky, kde jsou větší celky a jejich hodnocení [1]. Při použití této metody odhadování prochází tým seznam celků (vlastností, požadavků), které má vytvářet a každé položce v seznamu přiřadí velikost v podobě číselné hodnoty z jedné ze škál popsaných v tabulce níže.

Tabulka 12 – Nejběžnější měřítka větších celků [1]

Škála větších celků	**Konkrétní body na škále**
Mocniny čísla 2	1, 2, 4, 8, 16
Fibonacciho posloupnost	1, 2, 3, 5, 8, 13

Ačkoli je toto hodnocení bezrozměrné, neříká vlastně nic o tom kolik řádků kódu, počet dnů práce personálu, zkrátka nic srozumitelného, odhadl tým v jednom okamžiku všechny celky projektu, s použitím stejného měřítka a bez jakýchkoli předsudků. V další fázi je naplánována iterace, včetně naplánování dodání určitého počtu bodů větších celků. Po provedení iterace má tým určitou odhadovací schopnost, kdy může zjistit, kolik bodů větších celků dodal, kolik práce bylo potřeba a kolik času uplynulo. Z těchto výsledků lze provést počá-

teční kalibraci převodu větších celků na práci a čas. Data z počáteční kalibrace poté umožňují vytvořit odhad zbytku projektu. Tento postup však předpokládá, že pracovní tým zůstane v dalších iteracích stejný a nezapočítává možné absence jednotlivých pracovníků. Čím kratší iterace jsou, tím dříve se získají data použitelná pro odhad zbytku projektu a tím přesnější odhad bude.

2.5.4 T-shirt sizing

V této metodě, která se nazývá *konfekční velikost* [1], hodnotí každou vlastnost projektu dvě skupiny. V první skupině hodnotí vývojáři velikost, a tedy i cenu každé vlastnosti v měřítku od malé po velmi velkou. Zároveň je každá vlastnost ohodnocena zákazníkem, pracovníkem marketingu, obchodníkem nebo jinou pověřenou osobou měřítkem se stejnou škálou, vypovídajícím o důležitosti každé vlastnosti. Taková tabulka dokáže netechnickému investorovi jasně ukázat, které vlastnosti chce v projektu ponechat, a které z něj vyřadí. Například vlastnost, jejíž cena vývoje je příliš velká a její hodnota v projektu je malá. Takovéto rozpracování projektu dokáže již v rané fázi projektu protřídit nežádoucí vlastnosti a nemusí se tak nést dále do kužele nejistoty. K usnadnění rozhodnutí o tom, jakou vlastnost vzít a jakou vyhodit je také dobré seznam vlastností nahrubo setřídit podle poměru cena/výkon, což se obyčejně dělá přiřazením kombinované obchodní hodnoty. Jedná se o další bezrozměrné měřítko, které je postavené na kombinaci ceny vývoje a obchodní hodnoty pro projekt. Takové měřítko může vypadat následovně.

Tabulka 13 – Kombinované obchodní hodnoty [1]

	Cena vývoje			
Obchodní hodnota	Velmi velká (XL)	Velká (L)	Střední (M)	Malá (S)
Velmi velká (XL)	0	4	6	7
Velká (L)	-4	0	2	3
Střední (M)	-6	-2	0	1
Malá (S)	-7	-3	-1	0

Podle této tabulky je doplněna původní tabulka a seřazena od největších po nejmenších hodnot. Kombinovaná hodnota pomáhá rychlé vyhodnocení odpovědi typu „rozhodně ano“ nebo „rozhodně ne“.

3 PRINCIPY REGRESNÍCH MODELŮ

V této kapitole si uvedeme základní rozdělení regresních modelů. Regresní modely popisují vztah mezi závislou proměnnou *y* a nezávislou proměnnou *x*, kterých může být i více. Závislá proměnná *y* se nazývá response – odpověď, nebo také vysvětlovaná proměnná. Nezávislé proměnné *x* se nazývají vysvětlující proměnné nebo také prediktory.

Regresní model se snaží pomocí menšího počtu regresních koeficientů popsat vliv nezávislých proměnných (prediktorů) na závislé (vysvětlované) proměnné *y*. Takovým regresním modelem rozumíme model, jehož rovnice je ve tvaru (16):

$$y_i = f(x_{i1}, \dots, x_{il}; \beta_1, \dots, \beta_k) + \varepsilon_i, \qquad i = 1, \dots, n; j = 1, \dots, n \tag{16}$$

kde

$\boldsymbol{y_i}$	Závislá proměnná
$\boldsymbol{f}$	Funkce
$\boldsymbol{x_{ij}}$	Nezávislé proměnné = prediktory
$\boldsymbol{\beta_k}$	Regresní koeficienty
$\boldsymbol{\varepsilon_i}$	Náhodná chyba = reziduum

3.1 Lineární regrese

Lineární regrese je nejjednodušší typ regrese, díky čemuž je také nejpoužívanější. Modeluje vztah mezi závislou proměnnou *y* a jednou nebo více nezávislými prediktory *x*.

Jednoduchá lineární regrese [24] bere v úvahu k závislé proměnné *y* pouze jednu nezávisle proměnnou *x*. Tento vztah je při práci s reálnými daty rozšířen o náhodnou veličinu, která představuje odchylku od ideálního stavu (17). Této odchylce se říká reziduum a jedná se o rozdíl mezi pozorovanou a predikovanou hodnotou sledované veličiny.

$$y = \beta_0 + \beta_1 x + \varepsilon_i \tag{17}$$

kde

$\boldsymbol{\beta_0, \beta_1}$	Regresní koeficienty
$\boldsymbol{x}$	Prediktor
$\boldsymbol{\varepsilon}$	Reziduum, náhodná chyba

Rovnice lze zapsat také ve tvaru (18), kde ŷ je predikovaná (odhadovaná) hodnota.

$$y = \hat{y} + \varepsilon \tag{18}$$

$$\hat{y} = \beta_0 + \beta_1 x \tag{19}$$

V případě více prediktorů je vzorec rozšířen pro výpočet vícenásobného regresního modelu (17):

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i, \qquad i = 1, \ldots, n \tag{20}$$

kde

$\boldsymbol{\beta_p}$	Regresní koeficient, kde p=1,…,n
$\boldsymbol{x_{ip}}$	i-té pozorování prediktoru, kde p=1,…,n

Další, obecný, způsob zápisu vícenásobného lineárního modelu ve tvaru (18):

$$y_i = \beta_0 \sum_{k=1}^{K} \beta_k f_k(x_{i1}, x_{i2}, \ldots, x_{ip}) + \varepsilon_i, \;\; i = 1, \ldots, n \tag{21}$$

, kde f (.) je skalární funkce nezávislých proměnných x. Funkce f (x) mohou být v jakékoli formě, včetně nelineárních funkcí nebo polynomů. Linearita v lineárním regresním modelu odkazuje na linearitu koeficientů β_k. Jinými slovy, závislá proměnná y je lineární funkcí koeficientů β_k.

Maticový zápis rovnice [25] pro výpočet vícenásobného regresního modelu je ve tvaru (22):

$$y = X \times \beta + \varepsilon \tag{22}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ . \\ y_i \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & . & x_{1j} \\ x_{21} & x_{22} & . & x_{2j} \\ . & . & . & . \\ x_{i1} & x_{i2} & . & x_{ij} \end{pmatrix} \times \begin{pmatrix} \beta_1 \\ \beta_2 \\ . \\ \beta_j \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ . \\ \varepsilon_i \end{pmatrix} \tag{23}$$

, kde vektory y a ε mají rozměr $i \times 1$, vektor β má rozměr $j \times 1$ a matice X má rozměr $i \times j$.

Velké hodnoty reziduí značí špatnou schopnost modelu vysvětlit pomocí prediktorů cílovou proměnnou. Rovněž veškeré systematické chování prediktorů poukazuje na nevhodně nastavený model, především na špatný výběr prediktorů nebo jejich špatné specifikace. Některé proměnné je vhodnější zařadit do modelu namísto v lineární formě raději ve formě jiné, např. kvadratické, popřípadě proměnnou kategorizovat. Při kategorizaci proměnných je potřeba postupovat opatrně, s ohledem na to, že způsobená ztráta informace může zvýšit variabilitu modelu a docílit zkreslení výsledků.

Nejběžnější metodou pro výpočet odhadů parametrů β do regresních modelů je metoda nejmenších čtverců. Metoda poskytuje dostačující odhady parametrů při splnění všech předpokladů o datech a regresním modelu. V případě nesplnění všech předpokladů ztrácí výsledek své vlastnosti.

Předpoklady metody nejmenších čtverců [25]:

1. Regresní koeficienty β mohou nabývat libovolných hodnot. V praxi ovšem určitá omezení existují.
2. Regresní model je lineární v parametrech a platí aditivní model měření.
3. Matice nenáhodných, nastavovaných hodnot prediktorů x má hodnost rovnou právě m. Žádné dva sloupce tedy nejsou rovnoběžné vektory.
4. Náhodné chyby korelačních modelů mají vždy nulovou střední hodnotu $E(\varepsilon_i) = 0$. U regresních modelů se může stát, že střední hodnota $E(\varepsilon_i) = K$, kde i=1,…n. Takový model neobsahuje absolutní člen β_0. Po jeho zavedení bude platit rovnice $E(\varepsilon_i') = 0$, kde $\varepsilon_i' = y_i - \hat{y}_{P,i} - K$
5. Náhodné chyby mají konstantní a konečný rozptyl $E(\varepsilon_i) = \sigma^2$.
6. Náhodné chyby jsou vzájemně nekorelované. Chyby jsou nezávislé, pokud mají normální rozdělení.
7. Chyby ε_i mají normální rozdělení.

3.2 Kroková regrese

V případě více prediktorů je vhodné použití krokové regrese, která je kombinací sestupného a vzestupného výběru a spadá do skupiny sekvenčních postupů. V každém kroku je snaha o kombinaci vzestupného výběru s výběrem sestupným, s cílem zjednodušení modelu.

Postup krokové regrese lze popsat takto [27]:

1. Nastavení startovacího modelu s definovanými podmínkami
2. Nastavení parametrů finálního modelu - typ modelu, který je potřeba; typ podmínek
3. Nastavení mezní úrovně vyhodnocení
4. Po přidání nebo odebrání podmínky opakování testu modelu
5. Ukončení krokové regrese, pokud už nedochází k žádnému zlepšení odhadu

Postup u sestupného výběru je takový, že se nejprve spočítá nejbohatší model, a poté se začnou jednotlivé regresory z modelu postupně odebírat. Odebírá se vždy takový regresor, který je pro model nejméně užitečný. U vzestupného výběru je postup opačný. Začíná se

s prázdnou množinou regresorů, do které je v každém kroku přidán takový regresor, který v daném kroku nejlépe zlepšuje vysvětlení závislé proměnné. Kroková regrese vyžaduje dvě významové úrovně: první pro přidání proměnných a druhou pro jejich odstraňování. Pravděpodobnost omezení pro přidání proměnných by měla být menší než pravděpodobnost omezení pro odstranění proměnných, aby se krokový postup nedostal do nekonečné smyčky.

U regresního modelování, a především u krokové regrese, je doporučeno najít několik téměř optimálních modelů a ty poté mezi sebou porovnat a vybrat ten nejvhodnější.

II. PRAKTICKÁ ČÁST

4 CÍLE PRÁCE

Cílem této diplomové práce je navrhnout vhodný regresní model pro odhadování nákladů. Nyní, po uvedení do dané problematiky, máme potřebné předpoklady pro přejití k praktické části práce. Pro tyto účely jsou k dispozici data, která jsou získána ze tří různých zdrojů a obsahují proměnné používané pro odhad pomocí metody Use Case Points.

Na začátku odhadování je důležité ujasnit si, co přesně chceme odhadovat. Jestli celkovou velikost projektu, délku vývoje projektu nebo finanční náklady spojené s vývojem. Možností je několik, jak bylo uvedeno již v teoretické části. My máme v datech k dispozici, kromě hodnot jednotlivých aktérů a případů užití, také proměnné, které z těchto hodnot vychází, a sice hodnoty neupravených vah aktérů a případů užití, faktory prostředí, technické faktory a výslednou hodnotu UCP, která je poté ještě vynásobena faktorem produktivity. Faktor produktivity je podle Karnera stanoven na hodnotu 20 hodin na jeden bod případu užití. Bližšímu popisu samotných dat je věnována následující kapitola. Data budou pro tvorbu modelů rozdělena na dva menší datasety, poprvé pro surová data z UCP, tedy z jednotlivých aktérů a případů užití, a podruhé z proměnných pro výpočet UCP, tedy z neupravených vah aktérů a případů užití, z technického faktoru a faktoru prostředí. Pro nezávislé otestování vytvořených modelů budou ještě data rozdělena na trénovací a testovací část.

Nejprve bude vytvořen základní regresní model pomocí krokové regrese, který bude následně kalibrován s cílem dosažení co nejlepších výsledků na pozorovaných veličinách. Budou měněny startovací modely, ze kterých budou dané regresní modely vycházet, dále jejich kritéria a další podmínky, které mají vliv na výsledek regresního modelu. Poté budou jednotlivé regresní modely vyhodnoceny a porovnány podle základních kritérií – R^2, SSE, MSE, RMSE a AICc. Výběr nejlepších regresních modelů bude proveden na základě nejmenší chyby SSE, vypočtené z testovací části dat. Tyto dva modely budou následně vyhodnoceny a srovnány s výsledky z UCP. Smyslem práce je dojít k takovému regresnímu modelu, který bude vykazovat co nejlepší výsledky odhadu.

5 POPIS PRACOVNÍCH DAT

Dataset pracovních dat obsahuje prvky z Use Case Points – aktéry, případy užití, neupravené váhy aktérů a případů užití, technický faktor, faktor prostředí a údaj o skutečném počtu úsilí lidských hodin (Tabulka 14).

Tabulka 14 – Seznam proměnných v pracovních datech

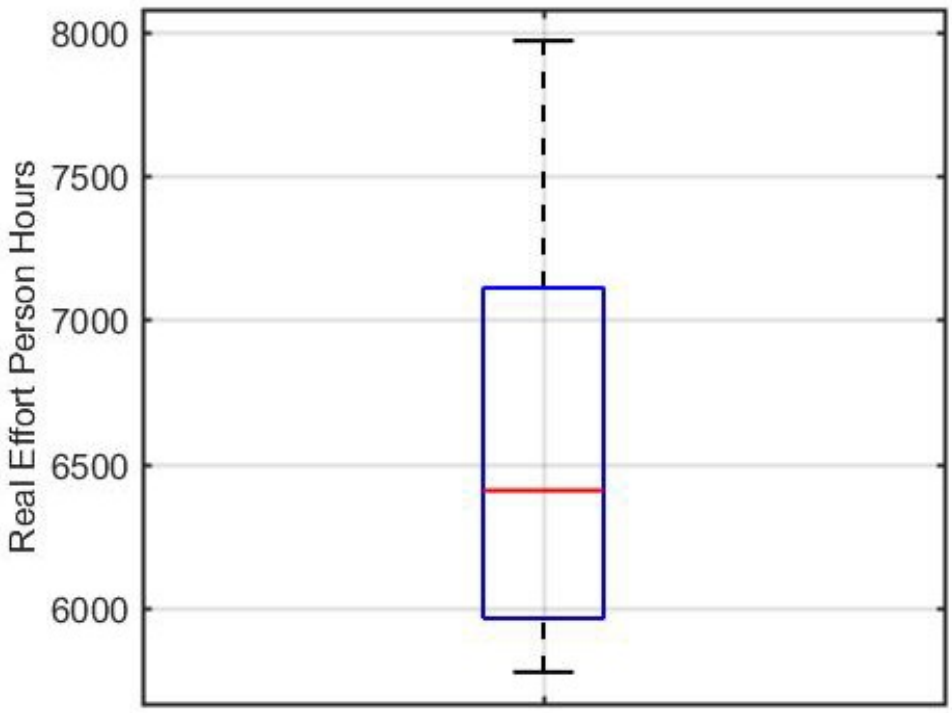
Název proměnné	**Popis**
Simple Actor	Jednoduchý aktér
Average Actor	Průměrný aktér
Complex Actor	Komplexní aktér
Simple UC	Jednoduchý případ užití
Average UC	Průměrný případ užití
Complex UC	Komplexní případ užití
UAW	Neupravené váhy aktérů
UUCW	Neupravené váhy případů užití
TCF	Technická složitost faktorů
ECF	Faktor prostředí
Real Effort Person Hours	Skutečný počet úsilí lidských hodin

V praktické části budeme kalibrovat regresní modely za účelem nalezení takového modelu, který by dosahoval nejlepších výsledků při odhadování skutečného počtu úsilí lidských hodin strávených na projektu. Popisná statistika proměnné Real_Effor_Person_Hours (Obr. 4) zobrazuje počet prvků v celém datasetu, průměrnou hodnotu, medián, minimální a maximální hodnotu a směrodatnou odchylku.

	Popisné statistiky (dataset)					
Proměnná	N platných	Průměr	Medián	Minimum	Maximum	Sm.odch.
Real_Effort_Person_Hours	70	6558,729	6406	5775	7970	664,2367

Obr. 4 – Popisná statistika reálného úsilí

Na krabicovém grafu pro reálné úsilí (Obr. 5) červená čára zobrazuje medián proměnné, který je podle popisné statistiky 6406. Minimální hodnota je 5775 a maximální je 7970.



Obr. 5 – Krabicový graf reálného úsilí

6 REGRESNÍ MODELY

Regresní modely jsou vytvořeny krokovou regresí pomocí funkce „stepwiselm“ v programu Matlab. Jedná se o systematickou metodu pro přidávání a odebírání termínů z lineárního nebo zobecněného lineárního modelu založeného na jeho statistické významnosti v odpovídající proměnné. Metoda začíná s počátečním, neboli startovacím modelem, který je určen pomocí *modelspec*, a pak porovnává po krocích větší a menší modely. Konečný model je vytvořen postupnou regresí, kdy na každém kroku metoda vyhledá podmínky založené na argumentu *Criterion* pro přidání nebo odebrání z modelu. Hodnoty pro kritéria jsou následující (Tabulka 15):

Tabulka 15 – Kritéria pro přidání nebo odebrání podmínek

Criterion	**Popis**
SSE	p-hodnota pro F-test změnu v součtu kvadratických chyb přidáním nebo odebráním výrazu
AIC	Změna hodnoty Akaikeho informačního kritéria
BIC	Změna hodnoty Bayesianova informačního kritéria
rsquared	Zvýšení hodnoty R^2
adjrsquared	Zvýšení hodnoty upravené R^2

Kromě toho jsou měněny také argumenty *Upper* a *Lower*. Argument *Upper* je specifikace modelu popisující největší množinu výrazů, *Lower* je argument pro specifikaci pojmů, které nemohou být odstraněny z modelu, tedy minimální podmínky.

V této práci budou vytvořené regresní modely vycházet ze tří startovacích modelů, a to z lineárního, kvadratického a čistě kvadratického. Podrobný přehled všech startovacích modelů v tabulce níže (Tabulka 16):

Tabulka 16 – Typy startovacích modelů krokové regrese

modelspec	**Popis**
Constant	Obsahuje pouze konstantní podmínky (intercept)
Linear	Obsahuje intercept a lineární podmínky pro každý prediktor

Interactions	Obsahuje intercept, lineární podmínky pro každý prediktor a všechny produkty prediktorů (bez čtvercových podmínek)
Purequadratic	Obsahuje intercept, lineární podmínky a čtvercové podmínky pro každý prediktor
Quadratic	Obsahuje intercept, lineární podmínky, interakce a čtvercové podmínky pro každý prediktor

Na začátku programu je načten soubor s pracovními daty. Pro účely naučení modelu a následného otestování jsou data rozdělena na trénovací a testovací část. To je zajištěno pomocí třídy „cvpartition“.

Základní syntaxe je:

```
c = cvpartition(group,'HoldOut',p)
```

kde

group	vynulovaný sloupec s čísly projektu ze souboru pracovních dat
p	skalární hodnota v intervalu $0 < p < 1$
‘HoldOut‘	způsob náhodného vytvoření validačního (testovacího) oddílu

Hodnotu *p* jsem nastavila na 0.3, díky čemuž jsem dostala z počtu původních pracovních dat dvě části dat, a sice testovací o velikosti 0.3 původních dat a trénovací část, která obsahuje zbylých 0.7 původních dat. K zajištění opakovatelného dosáhnutí stejného náhodného rozdělení dat byl na začátek programu dopsán příkaz „rng default“.

6.1 Kritéria hodnocení vytvořených regresních modelů

Vytvořené regresní modely budou vyhodnoceny nejprve pomocí kritérií R^2, SSE, MSE, RMSE a AICc. Nejlepší dva modely budou v další části podrobeny dalším hodnocením, na jejichž základě bude vybrán nejvhodnější model.

Koeficient determinace R^2

Hodnota R^2 vyjadřuje, jaká část z celkové variability ve výsledné závislé proměnné je vysvětlena pomocí vytvořeného regresního modelu [26]. Jsou dva typy hodnoty:

- Ordinary – obyčejná neupravená hodnota, která vyjadřuje poměr mezi hodnotami SSR a SST podle vzorce (24), kde SSR je regresní součet čtverců a SST je celkový součet čtverců

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} \tag{24}$$

- Adjusted – upravená hodnota podle počtu koeficientů. Je vhodnější než obyčejná neupravené hodnota, která je ovlivňována počtem vysvětlujících proměnných.

Výsledná hodnota se pohybuje v rozsahu od 0 do 1. Vynásobením této hodnoty číslem 100 lze hodnotu považovat za procento z celkové variability hodnot vysvětlované proměnné. Čím vyšší tato hodnota je, tím lepších výsledků regresní model dosahuje.

SSE – Sum of Squared Errors

Součet kvadratických chyb [26] je hodnota, která se vypočítá podle následujícího vzorce (25) a představuje součet rozdílů mezi skutečnými pozorováními a hodnotami pozorování předpovězených regresním modelem.

$$SSE = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \widehat{y_i})^2 \tag{25}$$

kde

$\boldsymbol{y_i}$	Skutečná (napozorovaná) hodnota
$\boldsymbol{\widehat{y_i}}$	Předpokládaná (odhadnutá) hodnota
$\boldsymbol{\varepsilon_i}$	Náhodná složka (chyba)
$\boldsymbol{n}$	Počet pozorování

Předpokládaná hodnota ŷ se vypočítá podle vzorce (19) uvedeného v kapitole lineární regrese.

MSE – Mean Squared Error

Průměrná kvadratická chyba [29] se vypočítá jako součet všech kvadratických chyb vydělený počtem pozorování (26). Čím více se hodnota MSE blíží nule, tím je kvalita odhadu lepší.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2 \tag{26}$$

RMSE – Root Mean Squared Error

Kořenová průměrná kvadratická chyba [29] je druhou odmocninou průměrné kvadratické chyby MSE a vypočítá se podle vzorce (27):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} \tag{27}$$

AIC (AICc)

Akaikeho informační kritérium [30] slouží k posouzení schopnosti různých modelů vysvětlit variabilitu v pozorovaných datech a vypočítá se podle vzorce (28):

$$AIC_i = -2 \times logL_i + 2 \times K_i \tag{28}$$

kde

$\boldsymbol{logL_i}$	Pravděpodobnost logaritmu
$\boldsymbol{K_i}$	Počet odhadovaných parametrů

Korigované Akaikeho informační kritérium je vhodné použít v případě malé variability. Bere do úvahy i velikost vzorku *n* a vypočítá se podle vzorce (29):

$$AIC_{Ci} = AIC_i + \frac{2 \times K_i \times (K_i + 1)}{n - K_i - 1} \tag{29}$$

Akaikeho kritérium zohledňuje kromě věrohodnosti modelu i jeho složitost. Modely s nižší hodnotou kritéria značí vhodnější model pro pozorovaná data. Modely se pomocí AIC porovnávají na základě rozdílu (30):

$$\Delta = AIC_i - AIC_{min} \tag{30}$$

MAPE – Mean Absolute Percentage Error

Průměrná absolutní procentuální chyba je dalším měřítkem přesnosti odhadu [31]. Pro každý bod pozorování je vypočten rozdíl mezi skutečnou a odhadovanou hodnotou a poté je tento

rozdíl znovu vydělen skutečnou hodnotou. Součet všech těchto absolutních chyb je nakonec vydělen počtem pozorování k získání průměrné hodnoty a vynásoben hodnotou 100 pro převedení na procenta (31). Opět platí, že čím nižší hodnota, tím je dosaženo lepších výsledků.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \widehat{y_i}|}{y_i} \times 100 \tag{31}$$

6.2 Vytvořené regresní modely

Kromě zmíněného rozdělení dat na trénovací a testovací část, jsou data rozdělena při tvorbě regresních modelů ještě na dvě skupiny, a sice podle jednotlivých proměnných. V prvním případě vychází regresní modely z hodnot UCP, tedy z proměnných UAW, UUCW, TCF a ECF, dále označovány jako **Data 1**. V dalším případě jsou regresní modely tvořeny ze surových hodnot jednotlivých aktérů a případů užití, dále pouze jako **Data 2**. V obou případech je jako závislá proměnná y, argument *ResponseVar*, použita proměnná Real_Effort_Person_Hours.

Model 1

První regresní model je vytvořen z výchozích hodnot všech argumentů, které budou v dalších modelech měněny. Startovací model „constant“ obsahuje pouze konstantní podmínky (intercept). Výchozí hodnota argumentu pro přidání nebo odebrání podmínek - *Criterion*, je „sse“, a znamená p-hodnotu pro F-test na změnu součtu čtvercových chyb přidáním nebo odebráním termínu. Výchozí hodnota pro *Lower* je „constant“ a výchozí hodnota *Upper* je „interactions“.

Konfigurace regresního modelu v programu Matlab ve tvaru:

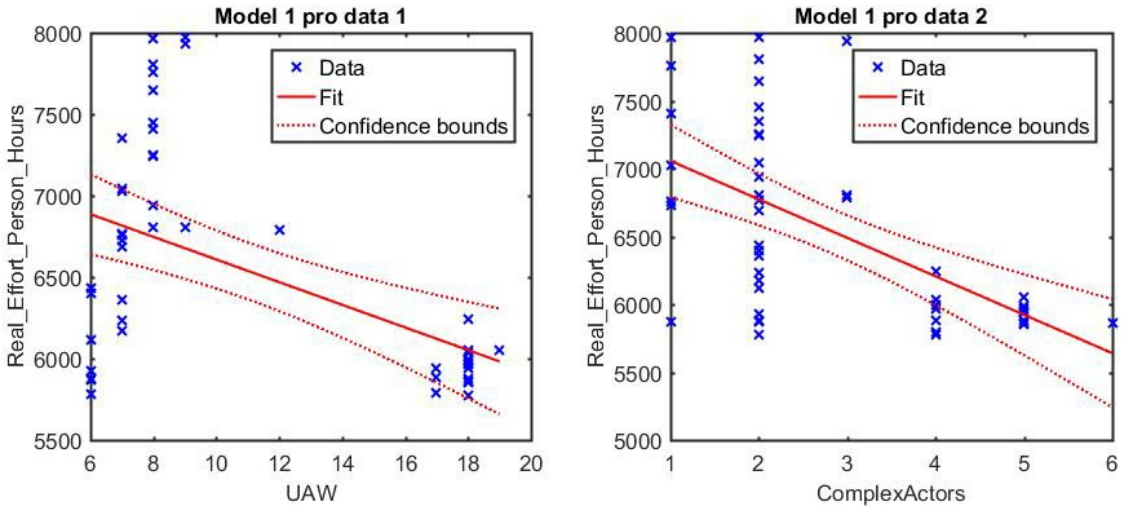
```
    mdl1 = stepwiselm(tbl,'constant','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','sse','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW
Data 2	Real_Effort_Person_Hours ~ 1 + ComplexActors

První graf je pro data 1, tedy z hodnot UCP (Obr. 6). Rovnice regresního modelu má jeden

prediktor, a to proměnnou UAW. Druhý graf je pro data 2, kde je rovněž pouze jeden prediktor, ComplexActors. Jelikož je v obou případech pouze jeden prediktor, je typ grafu bodový s proloženou křivkou a konfidenčními hranicemi a popisuje vliv každého jednoho prediktoru na závislou proměnnou Real_Effort_Person_Hours.



Obr. 6 – Graf modelu 1

Z tabulky hodnot (Tabulka 17) pro regresní model je patrné, že dosahuje lepších hodnot pro data 2, tedy s aktéry a případy užití. Míra naučenosti modelu je 0,3298, tedy skoro 33%.

Tabulka 17 – Srovnání pro model 1

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,2456	$1{,}7365*10^7$	$3{,}6947*10^5$	607,8388	769,4460
Data 2	0,3298	$1{,}5428*10^7$	$3{,}2825*10^5$	572,9290	463,6495

Model 2

Zatímco první model byl pouze konstantní bez nastavených podmínek, od druhého modelu se již začne s kalibrací a snahou získat co nejlepší možný výsledek. V prvním případě byl startovací model konstantní, nyní jsem jej v modelu 2 nastavila na lineární. Všechny další kritéria zůstávají prozatím ve výchozích hodnotách.

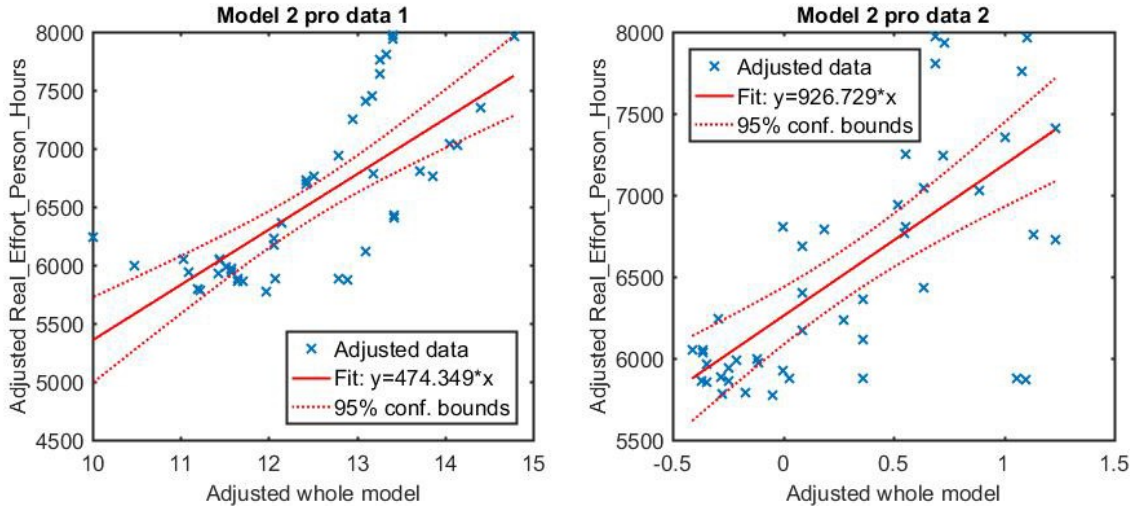
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl2 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','sse','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*UUCW
Data 2	Real_Effort_Person_Hours ~ 1 + ComplexActors + AverageActors*ComplexUC

Jak je vidět už podle výsledných formulí regresního modelu, v obou případech je více než jeden prediktor. Pro data 1 existují dva prediktory, k UAW přibyla ještě proměnná UUCW. Pro data 2 jsou prediktory tři, kromě komplexního aktéra, který byl i v základním modelu, je zde ještě průměrný aktér a komplexní případ užití. Grafy níže (Obr. 7) zobrazují vliv upravených nezávislých proměnných na upravenou závislou proměnnou.



Obr. 7 – Graf modelu 2

Hodnoty kritérií pro druhý regresní model vychází lépe pro data 1, pro které dosahuje míra naučenosti skoro 50% (Tabulka 18). Rovněž z rozložení dat pro jednotlivé datasety je vidět, že data pro první dataset jsou rozložena blíže k fitovací přímce.

Tabulka 18 – Srovnání pro model 2

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,4923	$1{,}1189*10^7$	$2{,}4864*10^5$	498,6392	752,5569
Data 2	0,4255	$1{,}2380*10^7$	$2{,}8135*10^5$	530,4281	759,9985

Model 3, 9 a 10

V dalším modelu jsem startovací model změnila z lineárního na kvadratický.

Konfigurace regresního modelu v programu Matlab ve tvaru:

```
    mdl3 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','sse','lower','constant','upper','interactions')
```

Stejných výsledků bylo dosaženo i ve dvou dalších modelech, a sice v modelu číslo 9 a 10. Pro úplnost tedy jen přikládám konfigurace těchto modelů.

Konfigurace regresního modelu 9 v programu Matlab ve tvaru:

```
    mdl9 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','bic','lower','constant','upper','interactions')
```

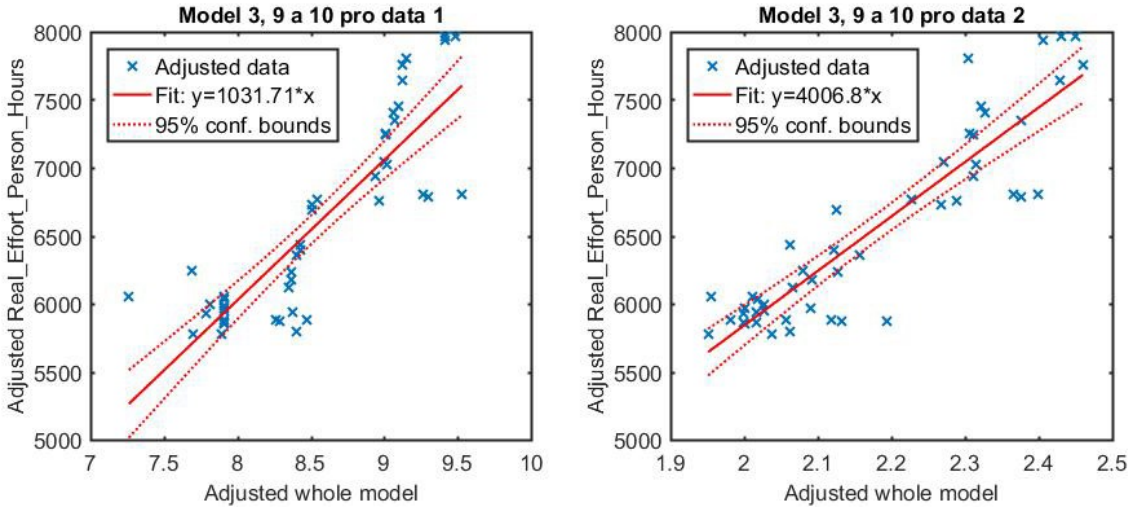
Konfigurace regresního modelu 10 v programu Matlab ve tvaru:

```
mdl10 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','bic','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UUCW + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*ComplexUC + AverageActors*ComplexActors + AverageUC*ComplexUC + AverageActors^2 + ComplexActors^2 + ComplexUC^2

Na základě formule regresního modelu je již patrné, že je model složitější, než modely předchozí. Model pro data 1 obsahuje opět dva intercepty, proměnné UAW a UUCW. Pro data 2 je jich již více – jednoduchý, průměrný a komplexní aktér a průměrný a komplexní případ užití. Na grafu níže (Obr. 8) je opět vidět, jak tyto prediktory ovlivňují výslednou závislou proměnnou. Rozložení dat je lepší v grafu pro data 2, data z projektu jsou soustředěná více kolem fitovací přímky v intervalech spolehlivosti.



Obr. 8 – Graf modelu 3, 9 a 10

Tabulka s hodnotami kritérií tento předpoklad potvrzuje (Tabulka 19). Ačkoli jsou výsledky pro oba datasety podobné, vhodnějším se jeví pro data 2. Míra naučenosti pro tento model se pohybuje v rozmezí 72 až 76%.

Tabulka 19 – Srovnání pro model 3, 9 a 10

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7232	$5{,}9654*10^6$	$1{,}3558*10^5$	368,2072	724,2247
Data 2	0,7605	$4{,}1059*10^6$	$1{,}1731*10^5$	342,5082	734,8788

Model 4

Další, poslední typ startovacího modelu, který budu při modelování využívat, je typ čistě kvadratický.

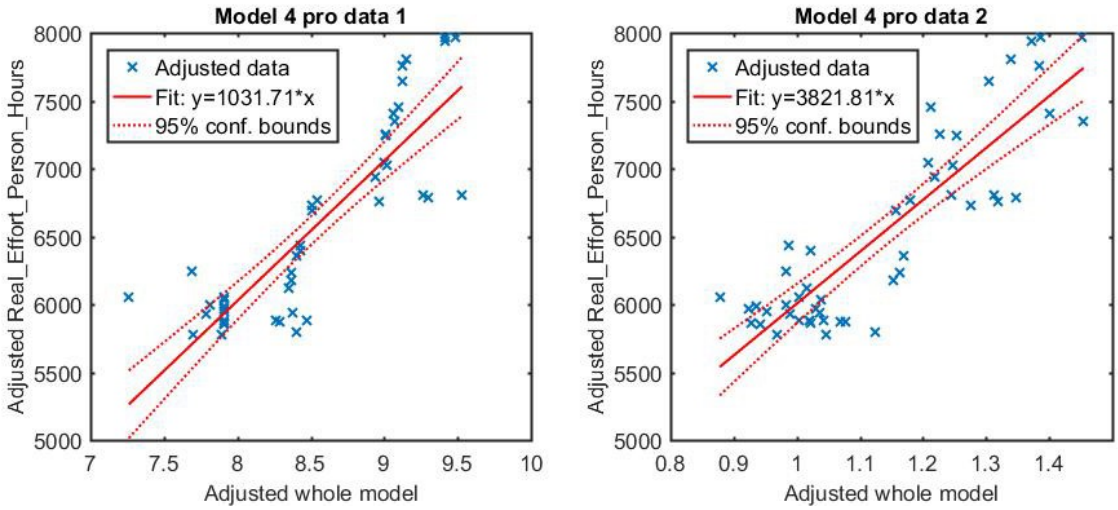
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl4 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','sse','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UUCW + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + SimpleActors*ComplexActors + AverageActors*ComplexActors + SimpleUC*AverageUC + ComplexActors^2 + AverageUC^2

Formule regresního modelu a výsledky sledovaných kritérií se pro čistě kvadratický startovací model neliší od předchozího modelu, který má jako startovací model nastaven kvadratický. Nepatrná změna ve výběru prediktorů je pro data 2, kde místo komplexního případu užití je v modelu obsažen jednoduchý případ užití (Obr. 9).



Obr. 9 – Graf modelu 4

Sledovaná kritéria mají pro oba datasety podobné výsledky (Tabulka 20), míra naučenosti modelu je ale nepatrně lepší pro data 1.

Tabulka 20 – Srovnání pro model 4

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7232	$5{,}9654*10^6$	$1{,}3558*10^5$	368,2072	724,2247
Data 2	0,7086	$5{,}2803*10^6$	$1{,}4271*10^5$	377,7720	739,5189

Model 5

V dalším testovacím modelu je nastaven startovací model opět na lineární. Poprvé je změněno i kritérium pro přidání dalších podmínek, z výchozího nastavení na změnu hodnoty Akaikeho informačního kritéria.

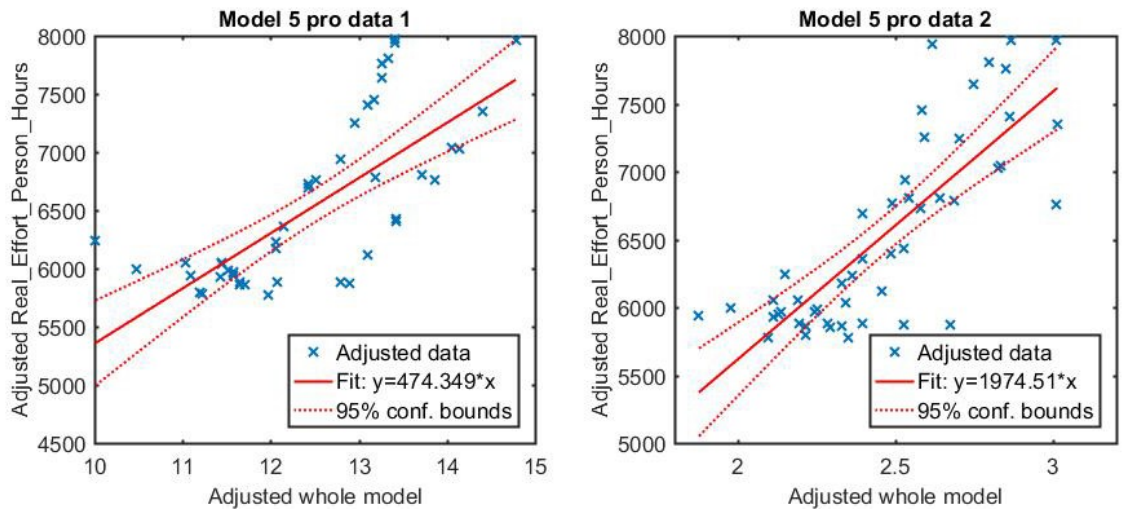
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl5 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','aic','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*UUCW
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + AverageActors*AverageUC + AverageActors*ComplexUC + ComplexActors*SimpleUC + ComplexActors*AverageUC + ComplexActors*ComplexUC

Zatímco pro data 1 jsou prediktory stále proměnné UAW a UUCW, pro data 2 jsou v tomto případě prediktory všechny varianty aktérů i případů užití. Fitovací přímka pro data 2 má kolmější průběh (Obr. 10) a rozložení projektů je soustředěné více kolem této přímky, než v případě grafu pro data 1.



Obr. 10 – Graf modelu 5

Srovnání v tabulce níže (Tabulka 21) ukazuje míru naučenosti modelu okolo 50% (od 49 do 52%).

Tabulka 21 – Srovnání pro model 5

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,4923	$1{,}1189*10^7$	$2{,}4864*10^5$	498,6398	752,5569
Data 2	0,5258	$8{,}3604*10^6$	$2{,}3223*10^5$	481,9046	765,7683

Model 6

V dalším modelu je opět změněn startovací model na kvadratický, kritérium zůstává nadále nastaveno na AIC.

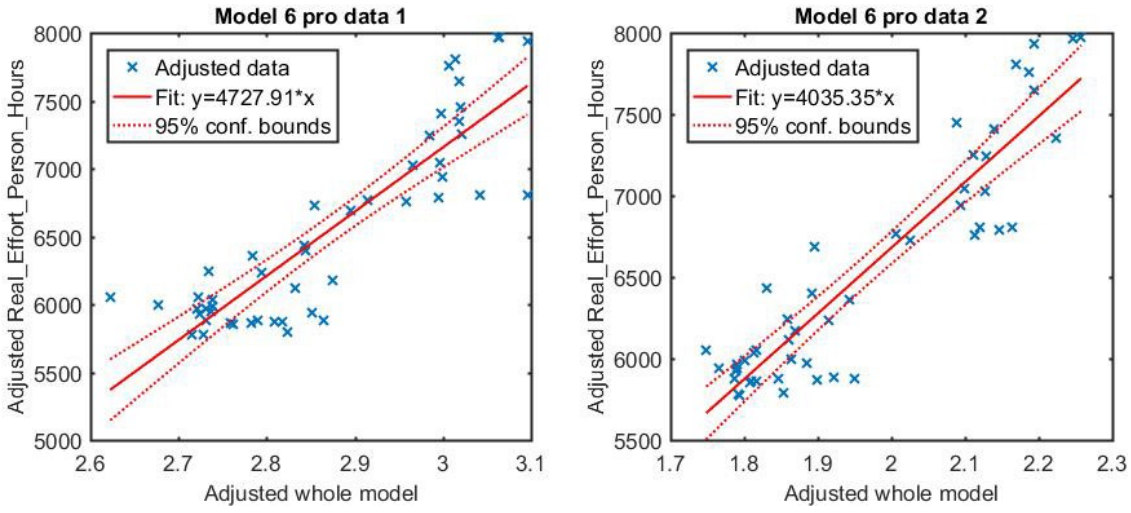
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl6 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','aic','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*TCF + UUCW*TCF + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*AverageUC + SimpleActors*ComplexUC + AverageActors*ComplexActors + ComplexActors*SimpleUC + AverageUC*ComplexUC + AverageActors^2 + ComplexActors^2 + ComplexUC^2

Poprvé jsou pro data 1 prediktory tři, kromě proměnných UAW a UUCW přibyla ještě proměnná technického faktoru TCF. Prediktory pro data 2 jsou stejné jako v modelu předchozím, všechny typy aktérů a případů užití. Z grafů pro jednotlivé data vychází lépe rozložení dat projektů ve druhém případě (Obr. 11). Všechna data projektů jsou zde blíže k fitovací přímce, na rozdíl od prvních dat, kde je několik bodů, které jsou od přímky vzdálenější.



Obr. 11 – Graf modelu 6

Tabulka kritérií ukazuje míru naučenosti modelů od 73 do 77% (Tabulka 22). Podle ostatních kritérií vychází lépe výsledky pro data 2.

Tabulka 22 – Srovnání pro model 6

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7339	$5{,}3429*10^6$	$1{,}3031*10^5$	360,9916	727,0296
Data 2	0,7771	$3{,}4935*10^6$	$1{,}0917*10^5$	330,4134	740,3536

Model 7

Další model vychází z čistě kvadratického startovacího modelu a opět z akaikeho informačního kritéria.

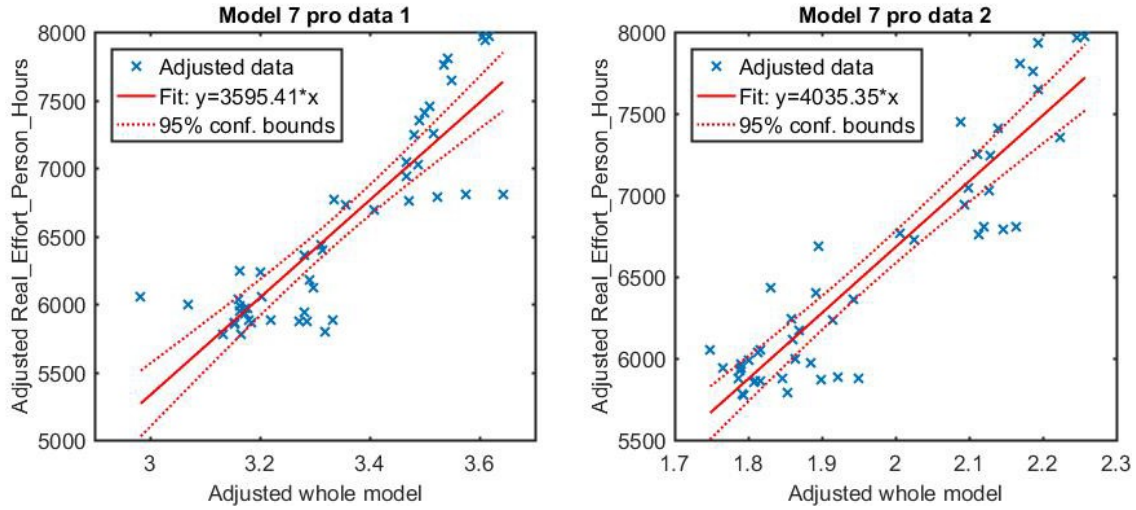
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl7 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','aic','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UUCW*ECF + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*AverageUC + SimpleActors*ComplexUC + AverageActors*ComplexActors + ComplexActors*SimpleUC + AverageUC*ComplexUC + AverageActors^2 + ComplexActors^2 + ComplexUC^2

Na rozdíl od kvadratického startovacího modelu, který dával prediktory UAW, UUCW a TCF, čistě kvadratický startovací model přináší prediktory UAW, UUCW a faktor prostředí ECF. Pro data 2 jsou prediktory opět všechny dostupné proměnné. Regresní model dosahuje pro data 2 stejných výsledků jako u předchozího modelu (Obr. 12). Pro data 1 se rozložení projektových dat nepatrně mění.



Obr. 12 – Graf modelu 7

Při srovnání výsledných hodnot daných kritérií pro model se startovacím modelem kvadratickým a čistě kvadratickým, jsou výsledky stejné (Tabulka 23). I zde je míra naučenosti modelu v rozsahu 73 až 77%.

Tabulka 23 – Srovnání pro model 7

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7332	$5{,}4883*10^6$	$1{,}3067*10^5$	361,4887	725,4769
Data 2	0,7771	$3{,}4935*10^6$	$1{,}0917*10^5$	330,4134	740,3536

Model 8

Model číslo 8 má nastaven lineární startovací model, a jeho kritérium pro přidání nebo odebrání podmínek je nastaveno na Bayesianovo informační kritérium.

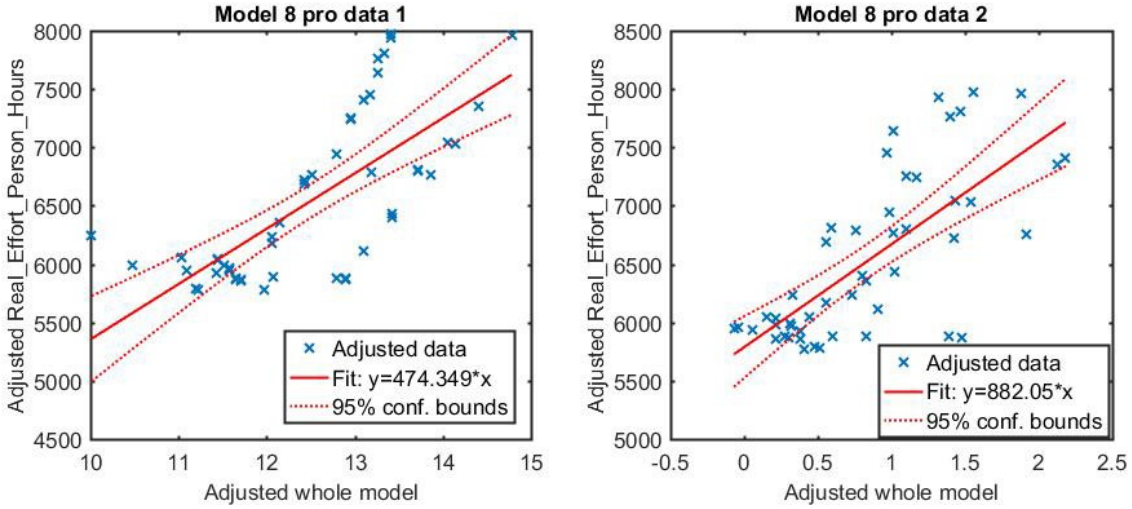
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl8 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Person_Hours','Criterion','bic','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*UUCW
Data 2	Real_Effort_Person_Hours ~ 1 + AverageActors*ComplexUC + ComplexActors*SimpleUC

Tento regresní model má pro data 1 prediktory proměnné UAW a UUCW, pro data 2 jsou prediktory průměrný a komplexní aktér a jednoduchý a komplexní případ užití (Obr. 13).



Obr. 13 – Graf modelu 8

I v tomto případě jsou hodnoty kritérií hodně podobné pro oba datasety (Tabulka 24), nicméně kromě SSE hodnoty dosahuje model lepších výsledků pro data 1.

Tabulka 24 – Srovnání pro model 8

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,4923	$1{,}1189*10^7$	$2{,}4864*10^5$	498,6392	752,5569
Data 2	0,4632	$1{,}1042*10^7$	$2{,}6289*10^5$	512,7326	759,7302

Model 11

Regresní model číslo 11 vychází opět z lineárního startovacího modelu a jako kritérium je nastaveno zvýšení determinačního koeficientu, hodnoty R^2.

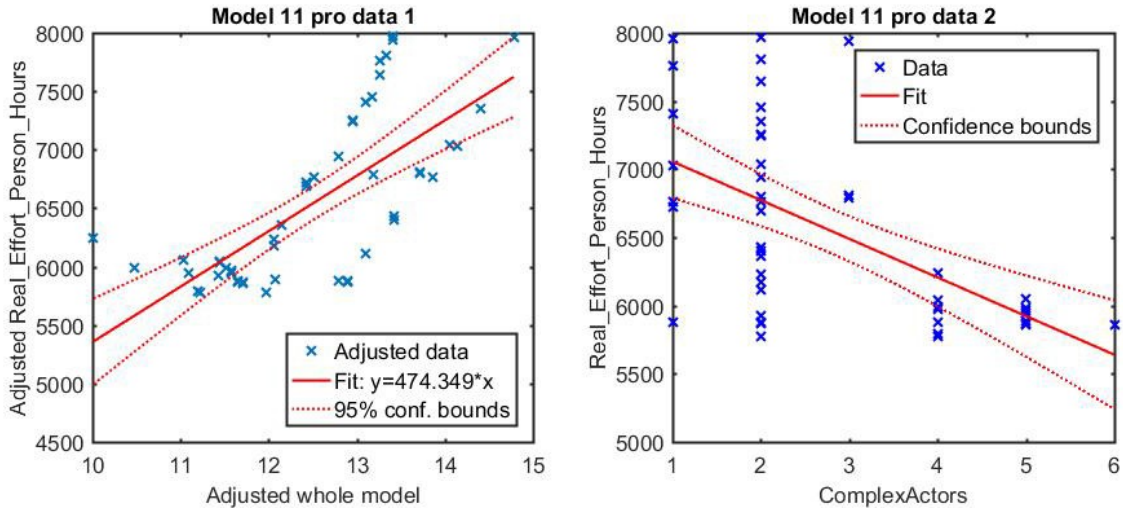
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl11 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Person_Hours','Criterion','rsquared','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*UUCW
Data 2	Real_Effort_Person_Hours ~ 1 + ComplexActors

V případě aplikování modelu na data 1 je formule regresního modelu ve stejném tvaru jako u modelů číslo 2, 5 a 8 a dostává tak stejné výsledky. V případě použití u dat 2 má formule pouze jeden prediktor, komplexního aktéra. Graf opět ukazuje vliv pouze jedné proměnné na výstup modelu (Obr. 14). Model dosahuje jednoznačně lepších výsledků pro data 1.



Obr. 14 – Graf modelu 11

Tabulka kritérií (Tabulka 25) toto tvrzení potvrzuje, pro data 1 je míra naučenosti necelých 50%, oproti 33% pro data 2. Rovněž ve všech ostatních kritériích dosahuje lepších hodnot.

Tabulka 25 – Srovnání pro model 11

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,4923	$1{,}1189*10^7$	$2{,}4864*10^5$	498,6392	752,5569
Data 2	0,3298	$1{,}5428*10^7$	$3{,}2825*10^5$	572,9290	763,6495

Model 12, model 13

Následující modely dosahují stejných výsledků, uvedu je zde tedy opět společně. V prvním případě je nastaven kvadratický startovací model a podruhé je startovací model čistě kvadratický. Kritérium pro přidání nebo odebrání podmínek zůstává nastaveno na zvýšení koeficientu determinace.

Konfigurace regresního modelu 12 v programu Matlab ve tvaru:

```
mdl12 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','rsquared','lower','constant','upper','interacti-
ons')
```

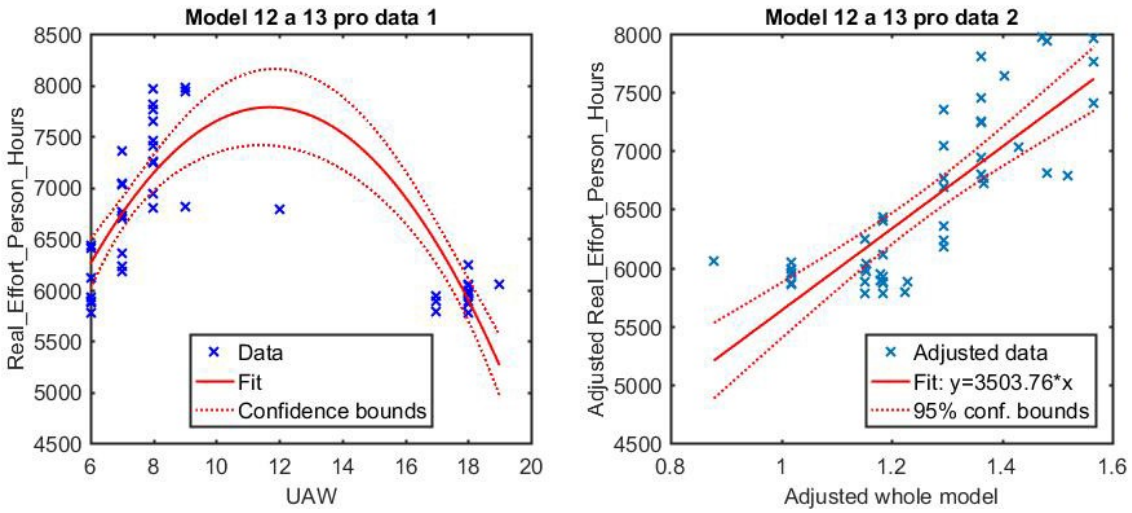
Konfigurace regresního modelu 13 v programu Matlab ve tvaru:

```
mdl13 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','rsquared','lower','constant','upper','interactions')
```

Formule regresních modelů zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UAW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*ComplexActors + AverageActors*ComplexActors + ComplexActors^2

Model pro data 1 obsahuje pouze jeden prediktor UAW, který je ovšem v tomto případě transformovaný a tvoří parabolu (Obr. 15). Pro data 2 existují tři prediktory – jednoduchý, průměrný a komplexní aktér. V obou případech jsou data projektů rozložena relativně blízko fitovací přímky, respektive křivky.



Obr. 15 – Graf modelu 12 a 13

Na základě jednotlivých kritérií dosahuje model ve všech kritérií lepších výsledků pro data 1, přestože je model pro data 2 téměř srovnatelný (Tabulka 26).

Tabulka 26 – Srovnání pro model 12 a 13

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,6468	$7{,}9566*10^6$	$1{,}7297*10^5$	415,8970	733,4764
Data 2	0,6108	$8{,}0050*10^6$	$1{,}9059*10^5$	436,5718	743,9717

Model 14

U regresního modelu 14 je nastaven lineární startovací model a kritérium je změněno na adjrsquared - zvýšení upraveného koeficientu determinace R^2.

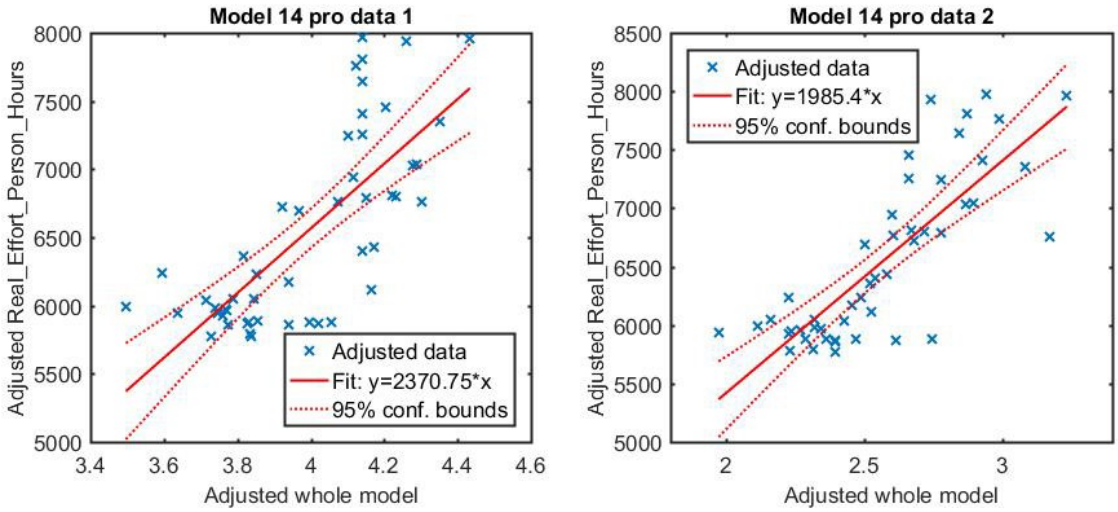
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl14 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Person_Hours','Criterion','adjrsquared','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + TCF + UAW*UUCW + UUCW*ECF
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + AverageActors*ComplexActors + AverageActors*AverageUC + AverageActors*ComplexUC + ComplexActors*SimpleUC + ComplexActors*AverageUC + ComplexActors*ComplexUC

Poprvé jsou jako prediktory využity pro oba datasety všechny jejich proměnné. Z grafů níže (Obr. 16) lze vyčíst lepší rozložení projektových dat pro druhý dataset – data 2, kde jsou data koncentrovány blíže k fitovací přímce.



Obr. 16 – Graf modelu 14

Z hlediska srovnání kritérií (Tabulka 27) je model pro obě skupiny dat podobný, lepších hodnot ale dosahuje na datech 2, čímž potvrzuje předpoklad z grafů.

Tabulka 27 - Srovnání pro model 14

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,4950	$1{,}0388*10^7$	$2{,}4734*10^5$	497,3339	756,7419
Data 2	0,5265	$8{,}1169*10^6$	$2{,}3191*10^5$	481,5717	768,2732

Model 15

Další model vznikne změnou startovacího modelu z lineárního na kvadratický. Kritérium zůstává nastaveno na zvýšení upraveného koeficientu determinace.

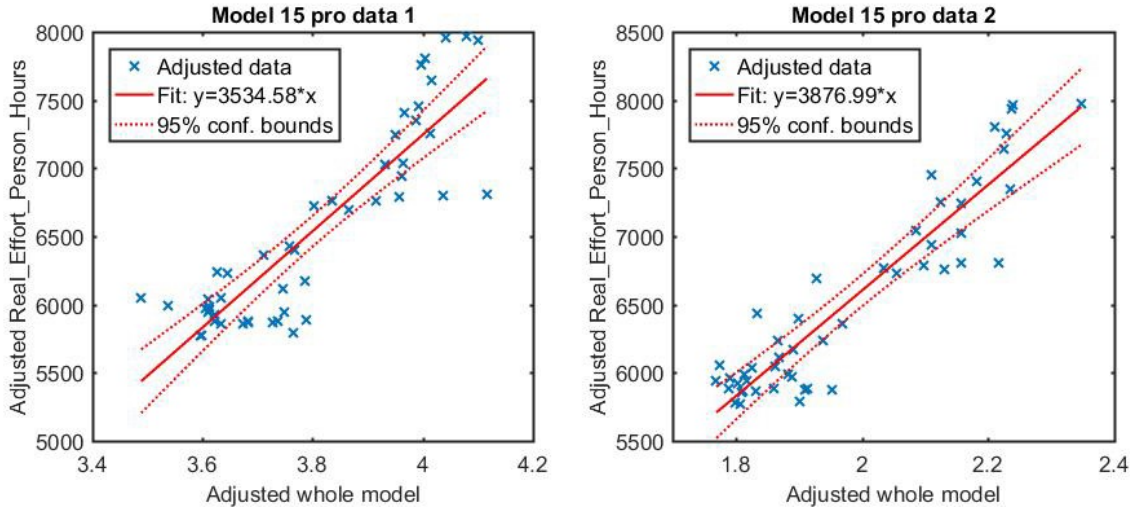
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl15 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','adjrsquared','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*UUCW + UAW*TCF + UAW*ECF + UUCW*TCF + UUCW*ECF + TCF*ECF + UAW^2 + UUCW^2 + TCF^2 + ECF^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*SimpleUC + SimpleActors*AverageUC + SimpleActors*ComplexUC + AverageActors*ComplexActors + AverageActors*SimpleUC + AverageActors*AverageUC + AverageActors*ComplexUC + ComplexActors*SimpleUC + ComplexActors*AverageUC + ComplexActors*ComplexUC + SimpleUC*AverageUC + SimpleUC*ComplexUC + AverageUC*ComplexUC + SimpleActors^2 + AverageActors^2 + ComplexActors^2 + SimpleUC^2 + AverageUC^2 + ComplexUC^2

Podle formulí regresního modelu uvedených výše, jde asi o nejsložitější výsledek regresního modelu. Při pohledu na grafické zobrazení modelů (Obr. 17) je vidět, že především pro data 2 jsou projektová data z velké části v intervalech spolehlivosti, rozložená kolem fitovací křivky. V ideálním případě by v těchto intervalech byly data všechny.



Obr. 17 – Graf modelu 15

Hodnoty sledovaných kritérií jsou pro obě data velmi podobné, v obou případech dosahuje míra naučenosti modelu bezmála 70% (Tabulka 28).

Tabulka 28 - Srovnání pro model 15

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,6893	$5{,}1743*10^6$	$1{,}5218*10^5$	390,1084	750,4035
Data 2	0,6890	$3{,}1984*10^6$	$1{,}5231*10^5$	390,2651	819,4873

Model 16

Regresní model s číslem 16 má startovací model čistě kvadratický, další nastavení zůstává stejné jako model předchozí.

Konfigurace regresního modelu v programu Matlab ve tvaru:

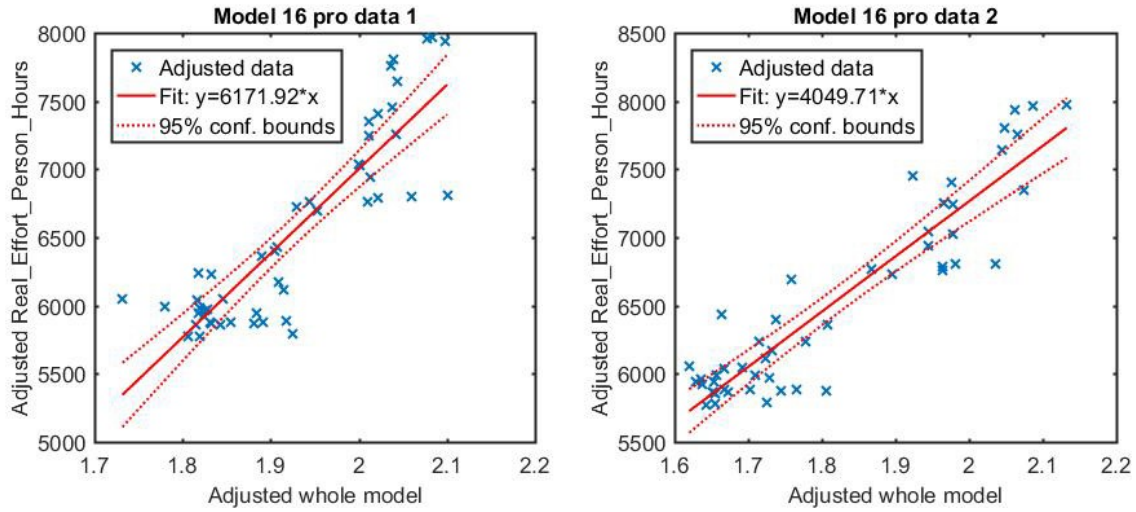
```
mdl16 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','adjrsquared','lower','constant','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + TCF + UUCW*ECF + UAW^2 + UUCW^2 + TCF^2 + ECF^2

Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*AverageUC + SimpleActors*ComplexUC + AverageActors*ComplexActors + SimpleUC*AverageUC + AverageUC*ComplexUC + SimpleActors^2 + AverageActors^2 + ComplexActors^2 + SimpleUC^2 + AverageUC^2 + ComplexUC^2

Sklon regresní křivky je pro data 1 strmější, než pro data 2 (Obr. 18). Rozložení projektových dat je v obou případech podobné, data se hlavně u spodní části soustředí kolem přímky v intervalech spolehlivosti.



Obr. 18 – Graf modelu 16

Hodnoty kritérií vychází opět velmi podobné (Tabulka 29), míra naučenosti modelu je pro oba datasety 77%. Podle SSE, MSE a RMSE dosahuje model lepších výsledků pro data 2, hodnota AICc je lepší pro data 1.

Tabulka 29 – Srovnání pro model 16

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7719	$5{,}3634*10^6$	$1{,}3752*10^5$	370,8394	733,4062
Data 2	0,7642	$3{,}3498*10^6$	$1{,}1551*10^5$	339,8698	754,5534

Model 17, model 19

Nyní, po vyčerpání možností základního nastavení, dojde poprvé ke změně parametru „upper“, tedy ke změně nastavení nejvyšší množiny výrazů. Regresní model bude vycházet z lineárního startovacího modelu, kritérium pro přidání nebo odebrání podmínek je nastaveno na Akaikeho informační kritérium a zmíněný parametr upper je nastaven na čistě kvadratický. Stejných výsledků dosahuje i regresní model číslo 19 se startovacím modelem čistě kvadratickým.

Konfigurace regresního modelu 17 v programu Matlab ve tvaru:

```
mdl17 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Person_Hours','Criterion','aic','lower','constant','upper','purequadratic')
```

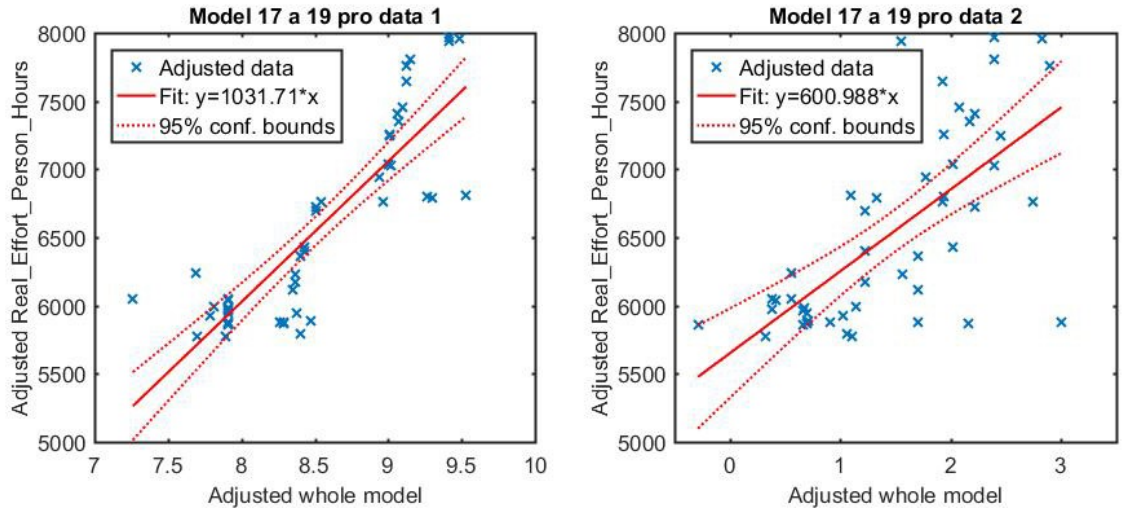
Konfigurace regresního modelu 19 v programu Matlab ve tvaru:

```
mdl19 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','aic','lower','constant','upper','purequadratic')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UUCW + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + AverageActors + ComplexActors + ComplexUC + AverageActors^2 + ComplexUC^2

Projektová data jsou pro první dataset soustředěná u regresní přímky (Obr. 19), pro druhý dataset jsou rozložená nesystematicky. Na první pohled je model pro první dataset vhodnější.



Obr. 19 – Graf modelu 17 a 19

Tento fakt potvrzuje tabulka s vypočtenými kritérii (Tabulka 30). Naučenost modelu pro data 1 je 72%, kdežto pro data 2 pouhých 40%. Rovněž ve všech dalších kritériích je model pro data 1 lepší.

Tabulka 30 – Srovnání pro model 17 a 19

Model	**R^2**	**SSE**	**MSE**	**RMSE**	**AICc**
Data 1	0,7232	$5{,}9654*10^6$	$1{,}3558*10^5$	368,2072	724,2247
Data 2	0,4092	$1{,}2443*10^7$	$2{,}8937*10^5$	537,9315	762,8532

Model 18

Regresní model 18 má startovací model kvadratický, veškeré další nastavené je stejné jako model číslo 17 a 19, tedy Akaikeho informační kritérium a čistě kvadratické nejvyšší možné množiny výrazů.

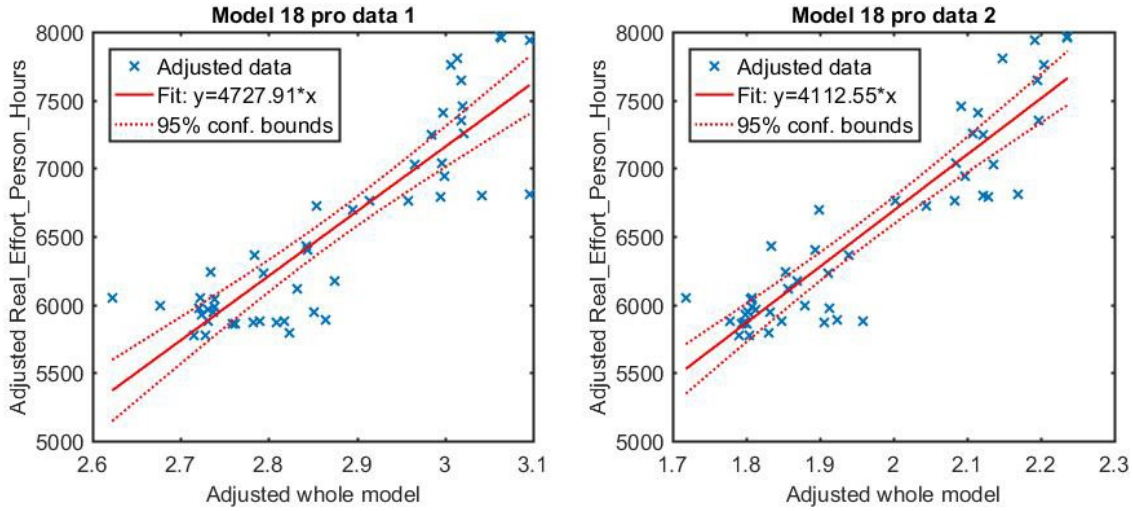
Konfigurace regresního modelu v programu Matlab ve tvaru:

```
mdl18 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','aic','lower','constant','upper','purequadratic')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW*TCF + UUCW*TCF + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleUC + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*AverageUC + SimpleActors*ComplexUC + AverageActors*ComplexActors + AverageUC*ComplexUC + AverageActors^2 + ComplexActors^2 + ComplexUC^2

Tento regresní model má, na rozdíl od předchozích dvou s lineárním a čistě kvadratickým startovacím modelem, pro oba datasety podobné výsledky (Obr. 20).



Obr. 20 – Graf modelu 18

Naučenost modelu je pro oba datasety více než 70%, lepších výsledků ale dosahuje model pro data 2, kde je míra naučenosti 77% (Tabulka 31).

Tabulka 31 – Srovnání pro model 18

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7339	$5{,}3429*10^6$	$1{,}3031*10^5$	360,9916	727,0296
Data 2	0,7722	$3{,}6811*10^6$	$1{,}1155*10^5$	333,9891	738,1743

Model 20

Regresní model číslo 20 vychází z lineárního startovacího modelu. Kritérium pro přidání nebo odebrání podmínky je nastaveno na zvýšení koeficientu determinace – rsquared. Parametr „upper“ je nastaven opět na jeho výchozí hodnotu, naopak parametr „lower“, tedy podmínky, které z modelu nemohou být odstraněny, je nastaven jako čistě kvadratický.

Konfigurace regresního modelu v programu Matlab ve tvaru:

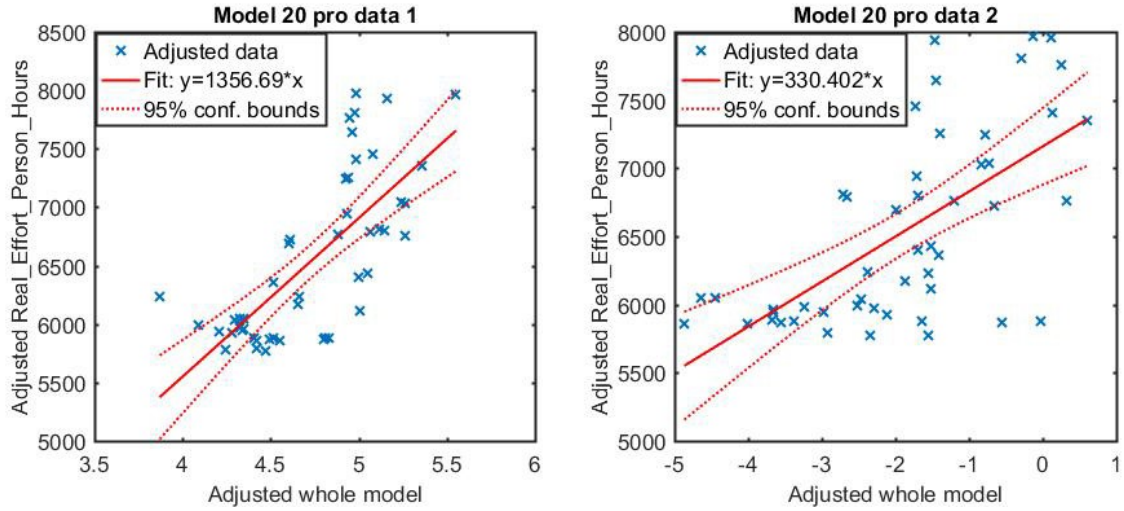
```
mdl20 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Person_Hours','Criterion','rsquared','lower','purequadratic','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + TCF + ECF + UAW*UUCW

Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors + AverageActors + ComplexActors + SimpleUC + AverageUC + ComplexUC

Regresní model využívá pro oba datasety k popisu vlivu nezávislých proměnných na reálné úsilí, všech svých proměnných jako svých prediktorů. Projektová data jsou lépe uspořádány kolem regresní křivky pro data 1 (Obr. 21).



Obr. 21 – Graf modelu 20

Regresní model vychází pro data 1 ve všech sledovaných kritériích lépe (Tabulka 32). Míra naučenosti dosahuje bezmála 50%.

Tabulka 32 – Srovnání pro model 20

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,4894	$1{,}0754*10^7$	$2{,}5010*10^5$	500,0982	755,7064
Data 2	0,3373	$1{,}3631*10^7$	$3{,}2455*10^5$	569,6912	770,0536

Model 21, model 22

V dalším regresním modelu je změněn startovací model na kvadratický, další nastavení zůstává stejné jako u předchozího modelu s číslem 20. Stejných výsledků dosahuje i model se startovacím modelem čistě kvadratickým.

Konfigurace regresního modelu 21 v programu Matlab ve tvaru:

```
mdl21 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','rsquared','lower','purequadratic','upper','interactions')
```

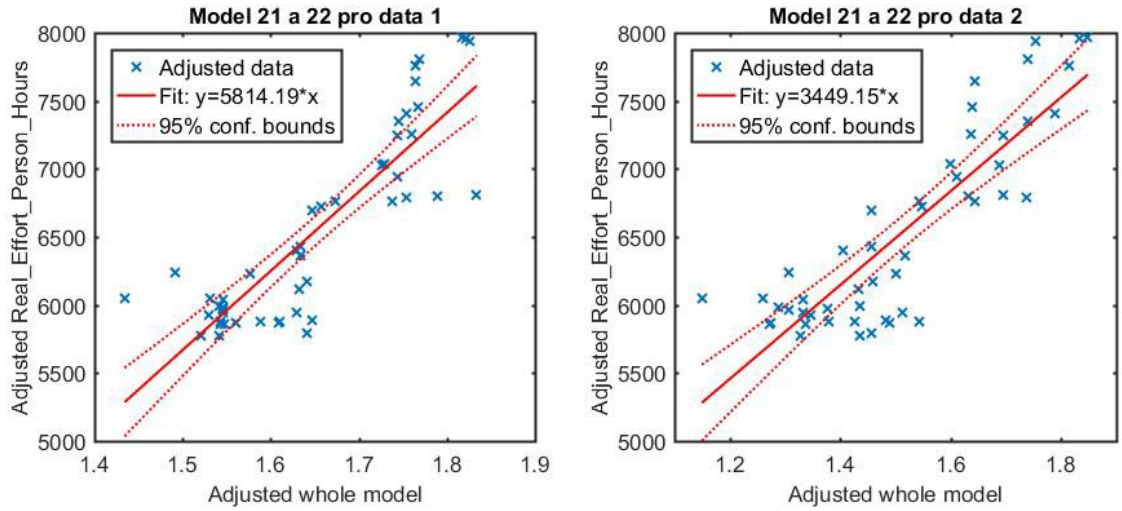
Konfigurace regresního modelu 22 v programu Matlab ve tvaru:

```
mdl22 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','rsquared','lower','purequadratic','upper','interactions')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UUCW + TCF + ECF + UAW^2 + UUCW^2 + TCF^2 + ECF^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleUC + AverageUC + ComplexUC + SimpleActors*ComplexActors + AverageActors*ComplexActors + SimpleActors^2 + AverageActors^2 + ComplexActors^2 + SimpleUC^2 + AverageUC^2 + ComplexUC^2

Model ke svému popisu opět využívá všech svých proměnných. Na grafu níže (Obr. 22) je vidět rozložení projektových dat pro oba datasety. Graf pro data 1 má data soustředěná více kolem fitovací přímky v hranicích spolehlivosti.



Obr. 22 – Graf modelu 21 a 22

Na základě sledovaných kritérií (Tabulka 33) dosahuje model lepších výsledků pro data 1.

Tabulka 33 – Srovnání pro model 21 a 22

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7086	$5{,}7097*10^6$	$1{,}4274*10^5$	377,8117	733,2981
Data 2	0,6473	$5{,}8735*10^6$	$1{,}7275*10^5$	415,6333	756,6146

Model 23, model 25

Další regresní model vychází z lineárního startovacího modelu. Jako kritérium pro přidání a odebrání podmínek je nastaveno Akaikeho informační kritérium, parametr „lower“ je nastaven na lineární podmínky, tedy minimální podmínky, které nemohou být z modelu odstraněny, a parametr ‚upper‘ je nastaven jako čistě kvadratický. Stejných výsledků dosahuje i model s čistě kvadratickým startovacím modelem.

Konfigurace regresního modelu 23 v programu Matlab ve tvaru:

```
    mdl23 = stepwiselm(tbl,'linear','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','aic','lower','linear','upper','purequadratic')
```

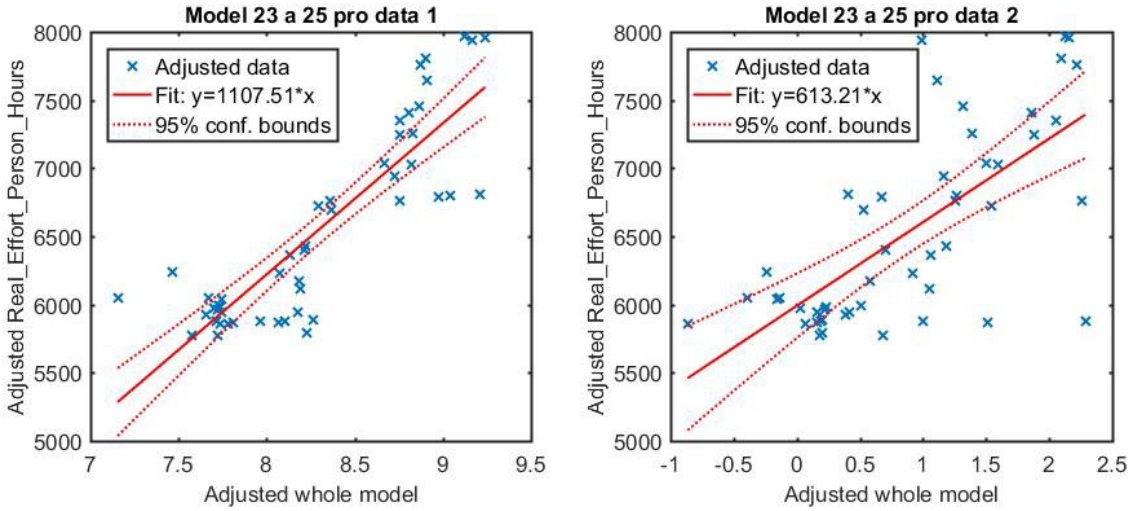
Konfigurace regresního modelu 25 v programu Matlab ve tvaru:

```
mdl25 = stepwiselm(tbl,'purequadratic','ResponseVar','Real_Effort_Per-
son_Hours','Criterion','aic','lower','linear','upper','purequadratic')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + UAW + UUCW + TCF + ECF + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleActors + AverageActors + ComplexActors + SimpleUC + AverageUC + ComplexUC + AverageActors^2 + ComplexUC^2

Na grafech níže (Obr. 23) je vidět, že jsou projektová data soustředěná více kolem fitovací křivky pro data 1, kde je jich hodně umístěno v hranicích spolehlivosti. Naopak pro data 2 jsou data rozprostřená více za touto hranicí.



Obr. 23 – Graf modelu 23 a 25

Tabulka kritérií ukazuje (Tabulka 34), že míra naučenosti modelu je pro první dataset 71%, pro druhý dataset pouze 39%. Rovněž všechny další kritéria vychází pro data 1 lépe.

Tabulka 34 – Srovnání pro model 23 a 25

Model	R^2	SSE	MSE	RMSE	AICc
Data 1	0,7156	$5{,}8499*10^6$	$1{,}3928*10^5$	373,2077	728,6035
Data 2	0,3916	$1{,}1918*10^7$	$2{,}9795*10^5$	545,8490	769,3568

Model 24

Poslední regresní model vychází z kvadratického startovacího modelu. Další natavení zůstává stejné jako u modelů číslo 23 a 25.

Konfigurace regresního modelu v programu Matlab ve tvaru:

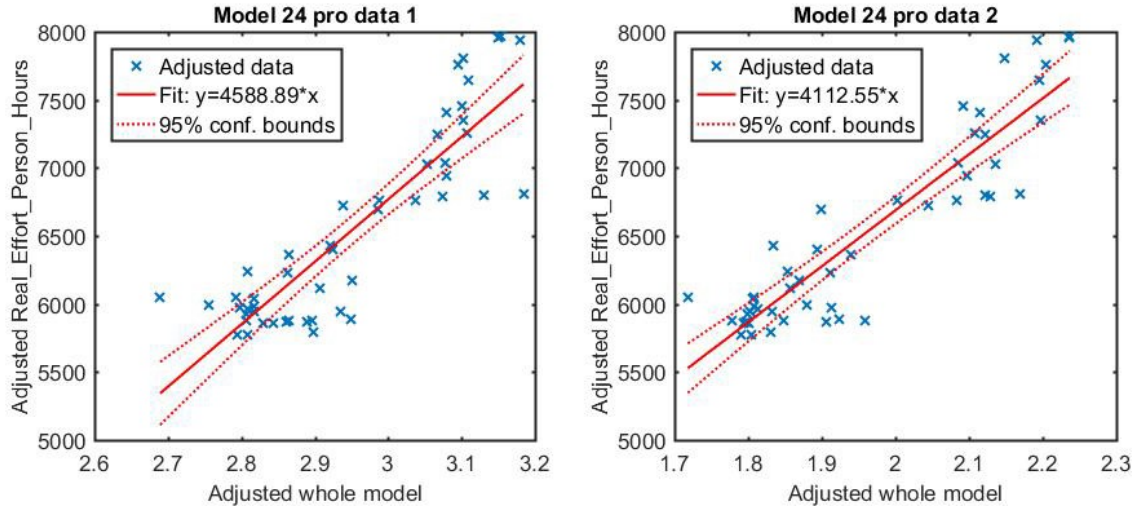
```
mdl24 = stepwiselm(tbl,'quadratic','ResponseVar','Real_Effort_Person_Hours','Criterion','aic','lower','linear','upper','purequadratic')
```

Formule regresního modelu zapsána ve Wilkinsonově notaci:

Data 1	Real_Effort_Person_Hours ~ 1 + ECF + UAW*TCF + UUCW*TCF + UAW^2 + UUCW^2
Data 2	Real_Effort_Person_Hours ~ 1 + SimpleUC + SimpleActors*AverageActors + SimpleActors*ComplexActors + SimpleActors*AverageUC + SimpleActors*ComplexUC +

	AverageActors*ComplexActors + AverageUC*ComplexUC + AverageActors^2 + ComplexActors^2 + ComplexUC^2

Vliv prediktorů na výstupní hodnotu je u tohoto modelu pro oba datasety velmi podobný. Data se v obou případech nachází jak v hranicích spolehlivosti kolem fitovací křivky, tak i za hranicemi (Obr. 24). V grafu modelu pro data 2 se ale data nenachází natolik vzdálená od hranic.



Obr. 24 – Graf modelu 24

Vypočtené hodnoty sledovaných kritérií ukazují, že je míra naučenosti opravdu velmi podobná (Tabulka 35). Pro data 1 dosahuje naučenost modelu 73%, pro data 2 je to 77%. Ve většině ostatních kritériích dosahuje model lepších výsledků rovněž pro data 2.

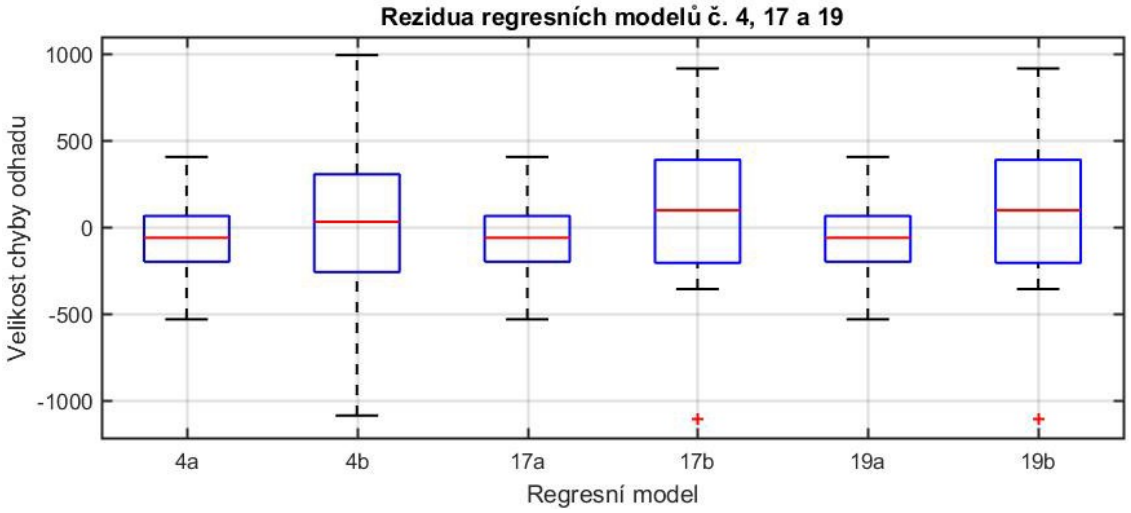
Tabulka 35 – Srovnání pro model 24

Model	**R^2**	**SSE**	**MSE**	**RMSE**	**AICc**
Data 1	0,7281	$5{,}3268*10^6$	$1{,}3317*10^5$	364,9233	729,8966
Data 2	0,7722	$3{,}6811*10^6$	$1{,}1155*10^5$	333,9891	738,1743

6.3 Srovnání regresních modelů podle SSE

Nyní jsem vytvořené regresní modely otestovala pomocí výpočtu SSE hodnoty na testovací části dat. Srovnání pro oba datasety je v tabulce níže (Tabulka 36). Na základě těchto výsledků jsem vybrala nejlepší model pro každý dataset. Jelikož hodnotíme podle SSE, hledáme nejnižší hodnotu.

Pro data 1 je nejlepší výsledek $1{,}0558*10^6$. Této hodnoty dosáhly modely číslo 3, 4, 9, 10, 17 a 19. Zaměřila jsem se tedy u těchto modelů na to, jak úspěšné byly pro druhou skupinu – data 2. Zde byla nejmenší hodnota $3{,}9996*10^6$ dosažena u modelů číslo 17 a 19, a hned v závěsu je model číslo 4 s hodnotou $4{,}2883*10^6$. Jelikož stále existuje několik možností a výsledek není zcela jasný, porovnala jsem rezidua těchto tří modelů – model 4, 17 a 19 (Obr. 25), kde je jako nejlepší možností jednoznačně model 4. Jako jediný z modelů nemá žádné odlehlé hodnoty. Označení regresních modelů na grafu níže je ve tvaru 4a pro data 1, model 4b pro data 2, atd.



Obr. 25 – Srovnání chyb regresních modelů č. 4, 17 a 19

Výběr nejlepšího regresního modelu pro data 2 proběhl na základě stejných kritérií. Nejlepší hodnota SSE je zde $2{,}4950*10^6$, které dosahují modely číslo 23 a 25. Vzhledem k tomu, že tyto dva modely dosahují naprosto stejných výsledků, jak bylo uvedeno již v předchozí kapitole v popisu jednotlivých regresních modelů, vybrala jsem model 23, který vychází z lineárního startovacího modelu.

Tabulka 36 – Srovnání SSE hodnot

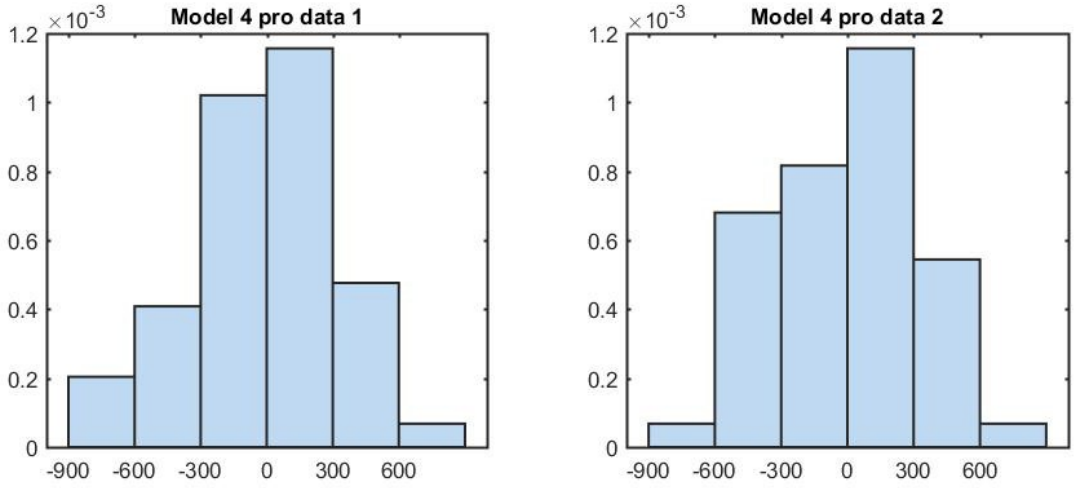
	Data 1	**Data 2**
Model 1	$4{,}6491*10^6$	$4{,}4681*10^6$
Model 2	$2.1847*10^6$	$5{,}0866*10^6$
Model 3	$1{,}0558*10^6$	$9{,}9074*10^6$
Model 4	$1{,}0558*10^6$	$4{,}2883*10^6$
Model 5	$2{,}1847*10^6$	$1{,}7656*10^7$
Model 6	$1{,}3290*10^6$	$1{,}0963*10^7$
Model 7	$1{,}5586*10^6$	$1{,}0963*10^7$
Model 8	$2{,}1847*10^6$	$1{,}7927*10^7$
Model 9	$1{,}0558*10^6$	$9{,}9074*10^6$
Model 10	$1{,}0558*10^6$	$9{,}9074*10^6$
Model 11	$2{,}1847*10^6$	$4{,}4681*10^6$
Model 12	$3{,}3503*10^6$	$1{,}0823*10^7$
Model 13	$3{,}3503*10^6$	$1{,}0823*10^7$
Model 14	$2{,}5589*10^6$	$8{,}8215*10^6$
Model 15	$1{,}7578*10^6$	$1{,}2740*10^7$
Model 16	$1{,}5643*10^6$	$2{,}6806*10^7$
Model 17	$1{,}0558*10^6$	$3{,}9996*10^6$
Model 18	$1{,}3290*10^6$	$1{,}4031*10^7$
Model 19	$1{,}0558*10^6$	$3{,}9996*10^6$
Model 20	$2{,}4878*10^6$	$2{,}6729*10^6$
Model 21	$1{,}2277*10^6$	$1{,}5360*10^7$
Model 22	$1{,}2277*10^6$	$1{,}5360*10^7$
Model 23	$1{,}0603*10^6$	$2{,}4950*10^6$
Model 24	$1{,}3584*10^6$	$1{,}4031*10^7$
Model 25	$1{,}0603*10^6$	$2{,}4950*10^6$

7 VYHODNOCENÍ

Tyto dva regresní modely, model 4 a model 23, nyní ve své další části diplomové práce porovnám spolu s výsledky z vypočtené UCP.

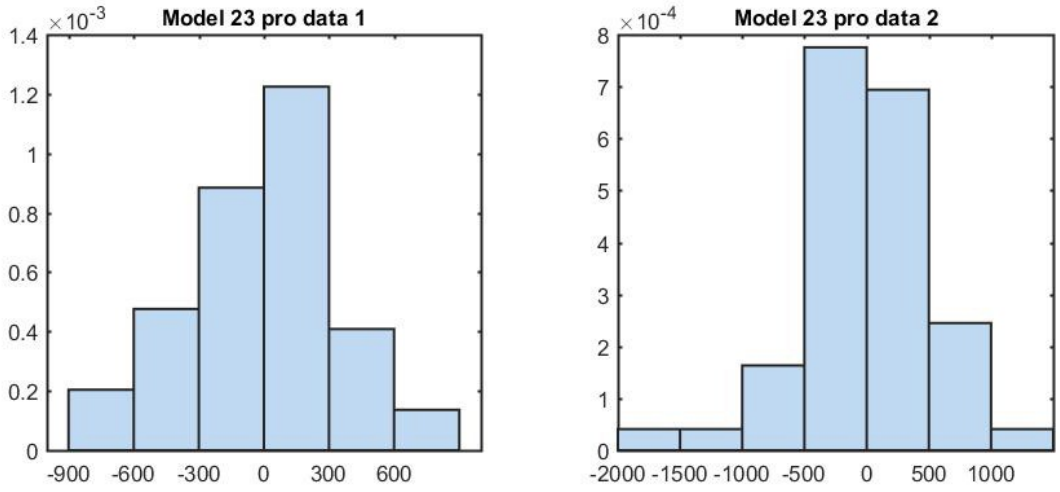
Rezidua regresních modelů

Funkce plotResiduals vrací grafické zobrazení reziduí v daném regresním modelu. Výchozí nastavení funkce je histogram, který lze použít k získání přehledu o distribuční hustotě vzorků z jedné proměnné. Zobrazuje tedy četnost zastoupení jednotlivých hodnot. Jedním z nejdůležitějších rozdělení je normální rozdělení, což je aspekt, který by měl být při analýze dat zohledněn. Histogram reziduí modelu 4 pro data 1 (Obr. 26) i histogram modelu 4 pro data 2 má normální rozdělení hodnot. Histogramy ukazují, že nejvíce hodnot má velikost rezidua v rozsahu od 0 do 300 a poté od -300 do 0. Rezidua jsou tedy centrována převážně kolem 0. Rozsah reziduí je pro obě data od -900 do 900.



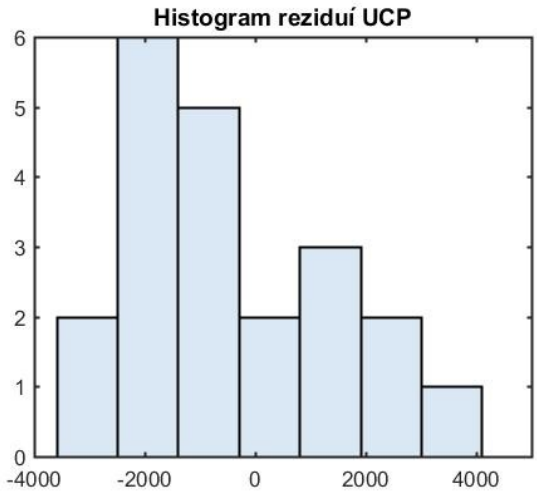
Obr. 26 – Histogram reziduí pro model 4

Histogram reziduí pro model 23 (Obr. 27) zobrazuje opět dva grafy, poprvé histogram pro data 1 a poté pro data 2. Histogram modelu 23 pro data 1 má rovněž normální rozdělení hodnot s rozsahem od -900 do 900. Rezidua modelu pro data 2 mají, oproti předchozím variantám, širší rozsah reziduí, od -2000 do 1500, díky čemuž se na první pohled zdá průběh rozložení jiný. Nicméně i zde se dá říci, že je rozložení reziduí normální.



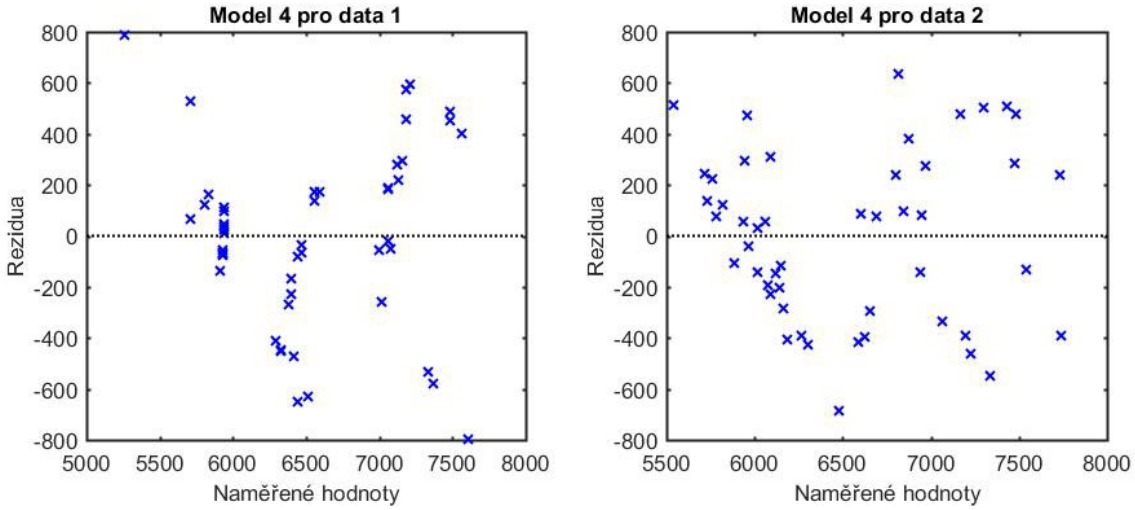
Obr. 27 – Histogram reziduí pro model 23

Pro srovnání, histogram reziduí u UCP (Obr. 28) má největší zastoupení hodnot okolo hodnoty -2000 a celkový rozsah reziduí má od -4000 do 4000.

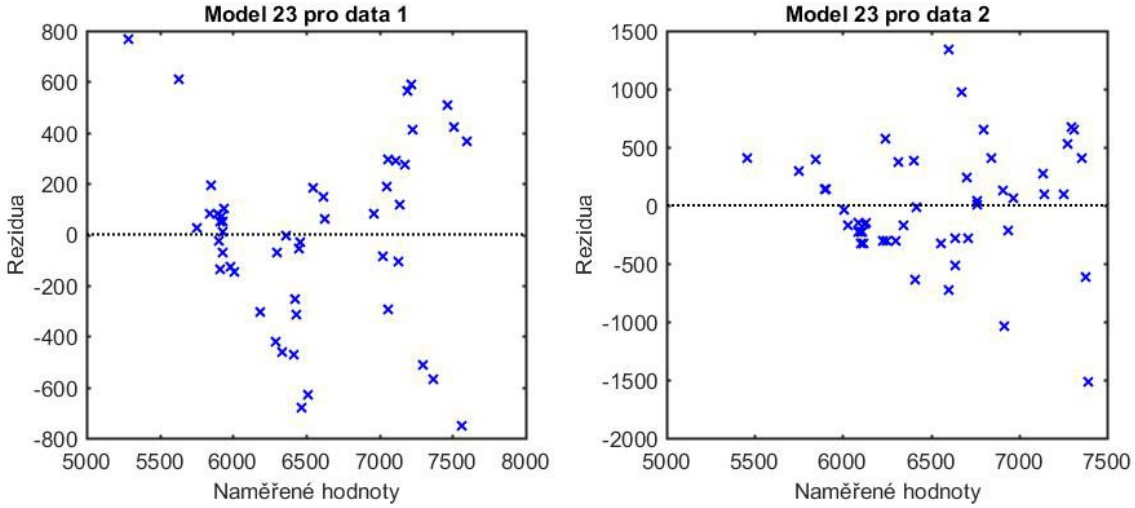


Obr. 28 – Histogram reziduí UCP

Další možnost, jak se podívat na rozložení reziduí, je závislost reziduí a naměřených hodnot. V ideálním případě nesmí graf vykazovat žádnou systematickou závislost v rozložení hodnot. Pokud jsou body náhodně rozmístěny kolem horizontální osy, je pro data vhodný lineární model. V opačném případě, kdy je rozložení bodů ve tvaru písmene U nebo obrácené U, je vhodnější model nelineární. V našem případě je rozložení bodů pro model 4 (Obr. 29) i pro model 23 (Obr. 30) rozmístěné kolem horizontální osy a tudíž v pořádku.



Obr. 29 – Rezidua modelu 4



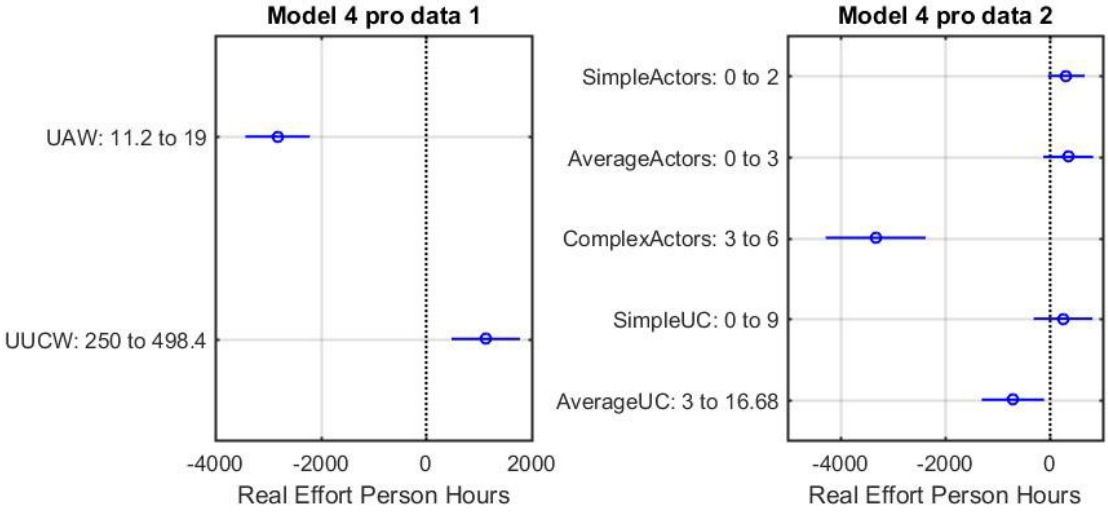
Obr. 30 – Rezidua modelu 23

Vliv prediktorů na výsledný model

Grafy plotEffects zobrazují přínos jednotlivých prediktorů na závislou proměnnou v regresním modelu, v našem případě tedy na reálné úsilí v člověkohodinách. Modré čáry zobrazují horní a dolní meze spolehlivosti pro závislou proměnnou. V ideálním případě tyto čáry zasahují pouze na jednu stranu od vertikální osy, čímž je jednoznačné, jaký má daný prediktor vliv na výslednou proměnnou.

Na obrázku níže (Obr. 31) vidíme grafy pro model 4. Pro data 1 má model č. 4 prediktory dva – UAW a UUCW. Graf nám ukazuje, že zvýšení prediktoru UAW z 11.2 na 19 vede ke snížení celkového počtu hodin o bezmála 3000 hodin. Zvýšení prediktoru UUCW z 250 na 498.4 vede naopak ke zvýšení pracnosti o přibližně 1000 hodin.

Pro stejné data má model č. 23 prediktory čtyři (Obr. 32). Vliv zvýšení hodnot TCF a ECF je velice blízko nule, efekt těchto dvou prediktorů je velmi malý a pro celkovou náročnost projektu jsou tyto prediktory nepodstatné. Prediktory UAW a UUCW mají ale stejný vliv jako u modelu 4. Zvýšení hodnoty UAW snižuje celkovou náročnost o cca 3000 hodin, zvýšení UUCW celkovou náročnost přibližně o 1000 hodin zvyšuje. Největší vliv má jednoznačně změna UAW. Z toho tedy vyplívá, že čím více aktérů v projektu je, tím se reálné úsilí snižuje.

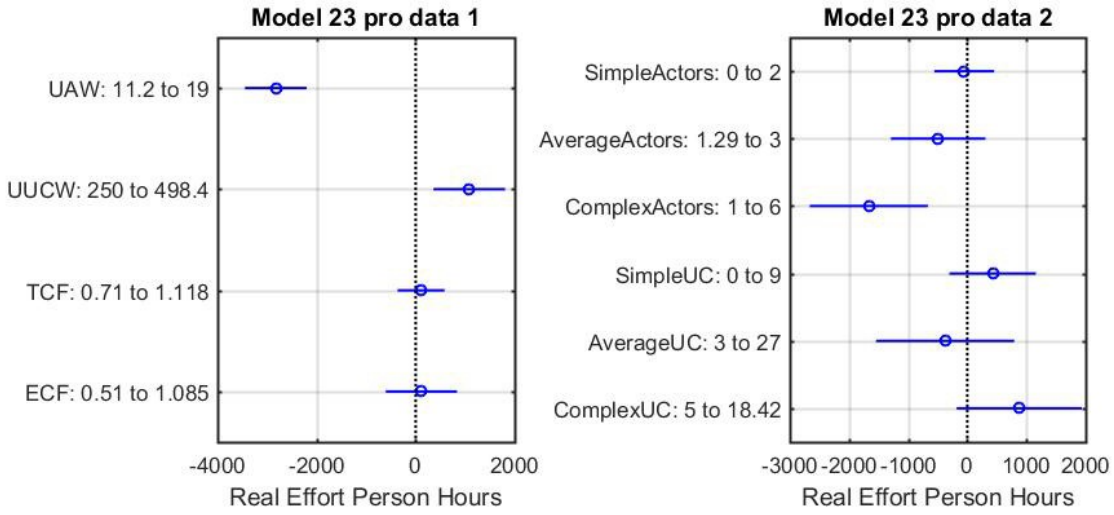


Obr. 31 – Vliv prediktorů na výstup pro model 4

Model č. 4 pro data 2 (Obr. 31) má pět prediktorů. Zvýšení jednoduchého aktéra z 0 na 2 má zanedbatelný vliv na výsledné celkové úsilí. Průměrný aktér má podobný vliv. Zvýšení jeho hodnoty z 0 na 3 má sice větší rozptyl mezí spolehlivosti, ale i tato hodnota se pohybuje těsně nad nulou. Největší vliv na celkové úsilí má komplexní aktér, jehož zvýšením z 3 na hodnotu 6 lze docílit snížení pracnosti o více než 3000 hodin. Zvýšení hodnoty jednoduchého případu užití má podobný vliv jako jednoduchý a průměrný aktér a je rovněž zanedbatelný. Zvýšení průměrného případu užití vede ke snížení pracnosti o necelý tisíc hodin. Po komplexním aktérovi je tedy druhým prediktorem, který má významnější vliv na výsledné úsilí.

Model č. 23 pro data 2 (Obr. 32) využívá k popisu modelu všech svých proměnných jako prediktorů. I zde má největší vliv změna hodnoty komplexního aktéra. Zvýšením jeho hodnoty z 1 na 6 se docílí snížení úsilí přibližně o 1600 hodin. Změna hodnoty jednoduchého aktéra nemá ani v tomto případě žádný vliv. Zvýšení průměrného aktéra vede, na rozdíl od modelu 4, ke snížení pracnosti přibližně o 500 hodin. K nárůstu celkového úsilí přispívá zvýšení jednoduchého a komplexního případu užití. V případě jednoduchého případu užití

je zvýšení přibližně o 500 hodin, zvýšení komplexního případu užití zvyšuje výsledek přibližně o 1000 hodin. Zvýšení hodnoty průměrného případu užití vede ke snížení pracnosti přibližně o 400 hodin, meze spolehlivosti však zasahují do obou stran od osy.



Obr. 32 – Vliv prediktorů na výstup pro model 23

Srovnání modelů s UCP

Závěrečné srovnání vybraných regresních modelů s UCP (Tabulka 37) je provedeno na testovacích datech podle hodnot průměrné kvadratické chyby MSE, kořenové průměrné kvadratické chyby RMSE, součtu kvadratických chyb SSE a průměrné absolutní procentuální chybě MAPE. Koeficient determinace R^2 je převzat ze srovnávacích tabulek kritérií pro jednotlivé modely z předchozí kapitoly, a je tedy jako jediný počítán z trénovacích dat. Dle koeficientu vychází lépe model 4, který dosáhl míry naučenosti nad 70% pro oba datasety. Výpočet tohoto kritéria pro UCP na hodnocení nemá význam.

Zaměříme-li se prozatím pouze na vybrané regresní modely, podle průměrné kvadratické chyby MSE je nejlepší model 4 pro data 1 (model 4a), jehož hodnota $5{,}0278*10^4$ je nejmenší, a tedy i jeho kořenová průměrná kvadratická chyba RMSE je nejmenší. Model 23 pro data 1 (model 23a) má průměrnou kvadratickou chybu MSE $5{,}0489*10^4$, tedy jen nepatrně vyšší, než model 4a. Pro data 2 mají oba modely průměrnou kvadratickou chybu o řád vyšší, model 4b s hodnotou $2{,}0421*10^5$ dopadl nejhůř, model 23b má hodnotu MSE $1{,}1881*10^5$.

Dopočítáním hodnot sledovaných kritérií pro UCP lze porovnat, o kolik jsou výsledky regresních modelů lepší. MSE a RMSE, vypočtené podle vzorců (26) a (27) uvedených v kapitole kritéria hodnocení, jsou pro UCP řádově vyšší. Průměrná kvadratická chyba pro UCP je $3{,}8693*10^6$ a jeho kořenová průměrná kvadratická chyba $1{,}697*10^3$.

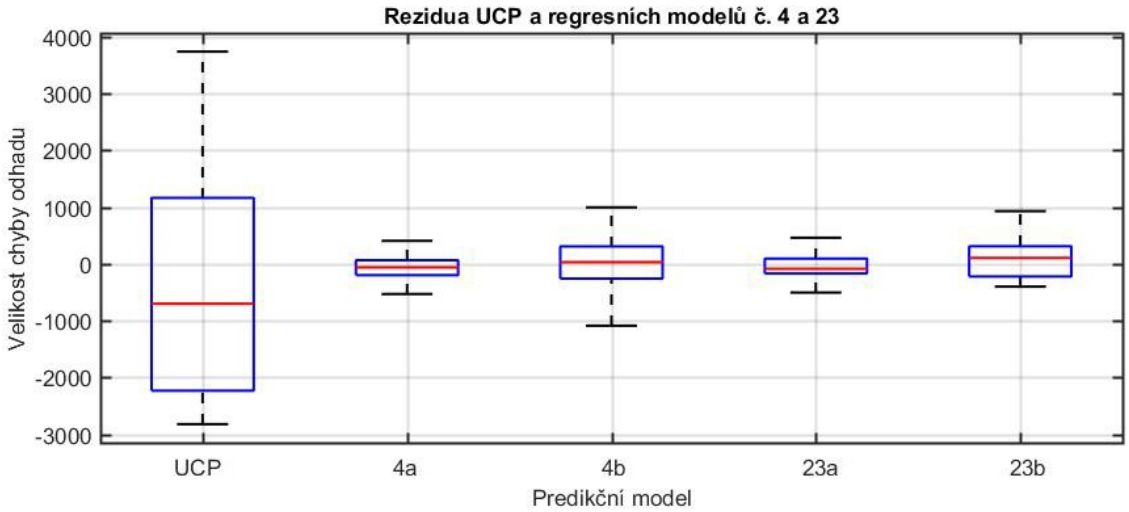
Tedy i nejhorší výsledky modelu 4b s hodnotami $2{,}0421*10^5$ pro MSE a 451,8919 pro RMSE jsou o dost lepší. Součet kvadratických chyb, vypočítaný podle vzorce (25), vychází $8{,}1255*10^7$, zatímco nejhorší výsledek u modelů je opět model 4b s hodnotou $4{,}2883*10^6$.

Pro všechny modely a UCP jsem dopočítala hodnotu MAPE podle vzorce (31). Výsledky průměrné absolutní procentuální chyby dobře demonstrují přesnost odhadů. Zatímco odhad pomocí UCP dosahoval chybovosti ve 26,82%, model 4a dosáhl nejlepšího výsledku s pouhými 2,69% chyb. Jedná se tedy o 10x lepší výsledek odhadu.

Tabulka 37 – Srovnání vybraných regresních modelů s UCP

Kritérium	UCP	Model 4a	Model 4b	Model 23a	Model 23b
R2	-	0,7232	0,7086	0,7156	0,3916
MSE	$3{,}8693*10^6$	$5{,}0278*10^4$	$2{,}0421*10^5$	$5{,}0489*10^4$	$1{,}1881*10^5$
RMSE	$1{,}9670*10^3$	224,2287	451,8918	224,6968	344,6891
SSE	$8{,}1255*10^7$	$1{,}0558*10^6$	$4{,}2883*10^6$	$1{,}0603*10^6$	$2{,}4950*10^6$
MAPE	26,8223	2,6901	5,2861	2,7656	4,3927

Na grafu níže (Obr. 33) jsou zobrazena rezidua UCP a regresních modelů. I zde je názorně vidět, o kolik jsou odhady provedené regresními modely přesnější oproti odhadu UCP.



Obr. 33 – Rezidua UCP a regresních modelů č. 4 a 23

Medián velikosti chyb pro odhad pomocí UCP je -697,5 (Tabulka 38), zatímco pro vybrané regresní modely se medián pohybuje v rozsahu -84,6 pro model 23a do 107,3 pro model 23b.

Tabulka 38 – Medián velikosti chyb predikčních modelů

	UCP	**Model 4a**	**Model 4b**	**Model 23a**	**Model 23b**
Medián	-697,552	-59,2473	33,0529	-84,6259	107,3708

Všechny vybrané modely dosahují na první pohled výrazně lepších výsledků než UCP. Při výběru nejlepšího regresního modelu na základě srovnání vybraných modelů (Tabulka 37), jsem zvolila nejlepším model 4a, který vychází z čistě kvadratického startovacího modelu, všechny ostatní kritéria má nastavené na výchozí hodnoty. Tento model dosahuje nejlepších výsledků a jeho průměrná absolutní chyba odhadu je pouhých 2,69%. Výběr tohoto modelu jako nejlepšího lze podepřít i krabicovým grafem reziduí UCP a regresních modelů (Obr. 33), kde rovněž model 4a vychází jako nejlepší s mediánem -59,2473. Model 23a dosahuje jen nepatrně horších výsledků než model 4a, průměrnou absolutní chybu odhadu má 2,7656, tedy jen o 0,0756% horší než model 4a, avšak medián jeho chyb je -84,6259, tedy opět horší.

Z těchto výsledků je ale vidět, že nejlepších odhadů dosahují modely pro data 1, s předzpracovanými hodnotami z UCP, tedy s UAW, UUCW, TCF a ECF. V závislosti na výsledcích diplomové práce se tedy jeví jako vhodnější používat pro odhady těchto hodnot, namísto aktérů a případů užití.

8 DALŠÍ MOŽNÝ ROZVOJ

Vývoj softwarového projektu je velmi kreativní proces, během kterého existuje celá řada faktorů, která ovlivňuje konečný výsledek [32]. Mohou se vyskytnout neočekávané problémy, může se s postupným vývojem přicházet na opomenuté funkce, vliv na konečný výsledek mohou mít i samotní vývojáři, kteří se na projektu podílí. Je obtížné vytvořit počáteční odhad natolik přesný, aby korespondoval s reálnou finální velikostí projektu, většina softwarových projektů se totiž v rané fázi potýká s nedostatkem podrobných informací. Je tudíž snaha tyto počáteční odhady zlepšit. K tomu slouží historická data, které by si měla každá firma uchovávat z již hotových projektů a pomoci nich poté snáze dosáhnout předpovědi projektu nového. Přestože jsou historická data velmi užitečným pomocníkem, kdy v počáteční fázi projektu lze s jejich pomocí pokrýt velké procento nevědomosti o projektu, díky jejich různorodosti a složení z rozdílných částí, nejsou metody odhadu schopny docílit optimálního odhadu.

Tento problém lze eliminovat shlukováním dat do skupin podle určitého přístupu. Smyslem rozdělení dat do těchto skupin je sdružovat podobná data na jednom místě. V mém případě bych mohla data shlukovat například podle stejného počtu aktérů a případů užití, zda se týkají stejné oblasti, podle druhu programovacího jazyka, atd. Jednou z možností, jak data rozčlenit do těchto skupin, je pomocí shlukovacího algoritmu. Shlukovacím algoritmem je například algoritmus K-means.

Shlukování pomocí algoritmu K-means patří do kategorie nehierarchických postupů [33], u kterého se předpokládá znalost požadovaného počtu shluků. V inicializaci se nastaví hodnota *k* vyjadřující počet shluků a dále počet bodů ve výchozí množině. Každý bod se poté přiřadí do shluku, jemuž se nejvíce podobá. V každém novém cyklu algoritmu se vypočítají nové středové hodnoty všech shluků jako algoritmické průměry všech jeho hodnot. Algoritmus pracuje tak dlouho, dokud nedosáhne co nejmenších rozdílů uvnitř všech shluků. Tímto postupem je zajištěna vysoká podobnost uvnitř každého shluku.

ZÁVĚR

Cílem diplomové práce bylo navrhnout vhodný regresní model pro odhadování nákladů. K modelování regresních modelů byl k dispozici soubor s daty obsahující proměnné používané pro odhad pomocí metody Use Case Points. K těmto datům jsem při modelování přistupovala dvěma způsoby. Data jsem rozdělila na dva přístupy, první dataset s koeficienty z UCP, a druhý dataset se zdrojovými hodnotami aktérů a případů užití. Dále byly data ještě rozděleny na dvě skupiny, na trénovací část pro naučení regresního modelu, a na testovací část pro jejich otestování a ověření nezávislosti výsledků na jiných datech.

Regresních modelů jsem postupnou kalibrací s cílem hledání nejlepších výsledků vytvořila celkem 25, z nichž některá nastavení dávala stejné výsledky, jak je uvedeno v detailním popisu v kapitole 6 Regresní modely. Při tvorbě modelů jsem každé nastavení podmínek pro výběr hodnot v dalších krocích regrese, měnila tři typy startovacích modelů – lineární, kvadratický a čistě kvadratický. Všechny tyto modely jsem porovnala na obou datasetech podle základních kritérií koeficientu determinace R^2, součtu kvadratických chyb SSE, průměrné kvadratické chybě MSE, kořenové průměrné kvadratické chybě RMSE a Akaikeho informačnímu kritérii AICc. Poté jsem všechny tyto modely otestovala na testovací části dat a porovnala podle součtu kvadratické chyby SSE (Tabulka 36). Na základě těchto výsledků jsem vybrala nejlepší model pro každý dataset.

Vybrané regresní modely č. 4 a 23 jsem podrobila dalšímu srovnání porovnáním jejich reziduí a vlivu prediktorů na výsledný odhad každého modelu. Na závěr jsem vybrané modely porovnala s UCP dle kritérií MSE, RMSE, SSE a průměrné absolutní procentuální chybě MAPE. Tato kritéria byla vypočtena na testovací části dat. Z výsledků (Tabulka 37) i z krabicového grafu reziduí jednotlivých predikčních modelů (Obr. 33) vyšel jako nejlepší model 4a, pro data s koeficienty z UCP. Regresní model vychází z čistě kvadratického startovacího modelu. Jeho průměrná absolutní procentuální chyba odhadu byla pouze 2,69%, zatímco při odhadu pomocí UCP byla chyba v odhadu 26,82%. Zlepšení v odhadu je tedy 10x lepší oproti odhadu UCP. Z výsledků mé práce vyplynulo, že je pro přesnější odhad vhodnější použít koeficienty z UCP, než počty jednotlivých aktérů a případů užití.

Další možný rozvoj regresních modelů pro zlepšení odhadování nákladů je rozdělení dat do tzv. shluků například pomocí shlukovacího algoritmu K-means, kdy se data postupně přiřadí k nejbližšímu shluku, kterému se podobá. Rozdělením dat do těchto shluků se získají skupiny dat, které si jsou podobnější. Odhad pomocí nich by tedy měl být přesnější.

SEZNAM POUŽITÉ LITERATURY

[1] MCCONNELL, Steve. *Odhadování softwarových projektů: jak správně určit rozpočet, termín a zdroje*. Brno: Computer Press, 2006. ISBN 80-251-1240-3.

[2] JONES, Capers. *Estimating software costs: bringing realism to estimating*. 2nd ed. New York: McGraw Hill, c1998. ISBN 00-791-3094-1.

[3] CONTE, Samuel Daniel, H. E. DUNSMORE a V. Y. SHEN. *Software engineering metrics and models*. Menlo Park, Calif.: Benjamin/Cummings Pub. Co., c1986. ISBN 08-053-2162-4.

[4] TRENDOWICZ, Adam a Ross JEFFERY. *Software project effort estimation: Foundations and Best Practice Guidelines for Success*. Switzerland: Springer International Publishing, 2014. ISBN 978-3-319-03628-1, 978-3-319-03629-8 (eBook).

[5] BROOKS, Frederick P. *The mythical man-month: essays on software engineering*. Reading, Mass.: Addison-Wesley Pub. Co., 1975. ISBN 02-010-0650-2.

[6] COSTELLO, Scott H. Software engineering under deadline pressure. *ACM SIGSOFT Software Engineering Notes* [online]. 1984, **9**(5), 15-19. DOI: 10.1145/1010941.1010947. ISSN 01635948. Dostupné z: http://portal.acm.org/citation.cfm?doid=1010941.1010947

[7] SACKMAN, H., W. J. ERIKSON a E. E. GRANT. Exploratory experimental studies comparing online and offline programming performance. *Communications of the ACM* [online]. 1968, **11**(1), 3-11. DOI: 10.1145/362851.362858. ISSN 00010782. Dostupné z: http://portal.acm.org/citation.cfm?doid=362851.362858

[8] WEINBERG, Gerald M. a Edward L. SCHULMAN. Goals and Performance in Computer Programming. *Human Factors: The Journal of the Human Factors and Ergonomics Society* [online]. 1974, **16**(1), 70-77. DOI: 10.1177/001872087401600108. ISSN 0018-7208. Dostupné z: http://hfs.sagepub.com/lookup/doi/10.1177/001872087401600108

[9] CURTIS, Christopher F. Possible methods of inhibiting or reversing the evolution of insecticide resistance in mosquitoes. *Pesticide Science* [online]. 1981, **12**(5), 557-564. DOI: 10.1002/ps.2780120513. ISSN 0031613x. Dostupné z: http://doi.wiley.com/10.1002/ps.2780120513

[10] BOEHM, Barry, Chris ABTS a Sunita CHULANI. Software development cost estimation approaches. *Annals of Software Engineering* [online]. 2000, **10**(1/4), 177-205. DOI: 10.1023/A:1018991717352. ISSN 10227091. Dostupné z: http://link.springer.com/10.1023/A:1018991717352

[11] PUTNAM, Lawrence H. a Ware. MYERS. *Measures for excellence: reliable software on time, within budget*. Englewood Cliffs, N.J.: Yourdon Press, 1992. ISBN 01-356-7694-0.

[12] HUMPHREY, Watts S. *A discipline for software engineering*. Reading, Mass.: Addison-Wesley, c1995. ISBN 02-015-4610-8.

[13] JØRGENSEN, M. A review of studies on expert estimation of software development effort. *Journal of Systems and Software* [online]. 2004, **70**(1-2), 37-60. DOI: 10.1016/S0164-1212(02)00156-5. ISSN 01641212. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0164121202001565

[14] LEDERER, Albert L. a Jayesh PRASAD. Nine management guidelines for better cost estimating. *Communications of the ACM* [online]. **35**(2), 51-59. DOI: 10.1145/129630.129632. ISSN 00010782. Dostupné z: http://portal.acm.org/citation.cfm?doid=129630.129632

[15] BOEHM, Barry W. *Software engineering economics*. Englewood Cliffs, N.J.: Prentice-Hall, c1981. ISBN 01-382-2122-7.

[16] BANERJEE, Gautam. *Use Case Points: -An Estimation Approach*. 2001. Dostupné z: http://www2.fiit.stuba.sk/~bielik/courses/msi-slov/reporty/use_case_points.pdf

[17] COHN, Mike. *Es timating With Use Case Points*. Dostupné také z: http://www.cs.cmu.edu/~jhm/DMS%202011/Presentations/Cohn%20-%20Estimating%20with%20Use%20Case%20Points_v2.pdf

[18] RIBU, Kirsten. *Estimating Object-Oriented Software Projects with Use Cases* [online]. Oslo, 2001. Dostupné také z: http://www.bfpug.com.br/Artigos/UCP/Ribu-Estimating_O-O_SW_Projects_with_Use_Cases.pdf

[19] MATHEW, Ruben Gerad. *Progressive Function Point Analaysis: Advanced Estimation Techniques for IT Projects* [online]. Amazon Digital Services LLC, 2014. ASIN: B00NH0MMAG. Dostupné také z: http://www.amazon.com/dp/B00NH0MMAG

[20] LONGSTREET, David. *Fundamentals of Function Point Analysis* [online]. 2005. Dostupné také z: http://www.softwaremetrics.com/files/Fundamentals%20of%20Function%20Point%20Analysis.pdf

[21] MENDES, Emilia. Cost estimation techniques for web projects. Hershey PA: IGI Pub., c2008. ISBN 978-159-9041-377. ISBN 978-1-59904-137-7 (eBook).

[22] KOIRALA, Shivprasad. How to prepare Software Quotation \matsymb{lbrack}online\matsymb{rbrack}. Dostupné z: http://webapps.6te.net/HowtoPrepareSoftwareQuotations.pdf

[23] GALORATH, Daniel D. a Michael W. EVANS. *Software sizing, estimation, and risk management: when performance is measured performance improves*. Boca Raton, FL: Auerbach Publications, 2006. ISBN 08-493-3593-0.

[24] KOMENDA, Martin. *Lineární regresní model* [online]. Dostupné také z: https://is.muni.cz/www/98951/41610771/43823411/43823458/Analyza_a_hodnoc/46097316/textVJ02_linreg.pdf

[25] MELOUN, Milan a Jiří MILITKÝ. *Statistická analýza experimentálních dat* [online]. Vyd. 2., upr. a rozš. Praha: Academia, 2004. ISBN 80-200-1254-0. Dostupné také z: https://meloun.upce.cz/docs/books/ucebnice-sken.pdf

[26] ŠMILAUER, Petr. *Moderní regresní metody* [online]. České Budějovice, Biologická fakulta JU, 1998-2007. Dostupné také z: http://regent.jcu.cz/MRM.pdf

[27] SILHAVY, Radek; SILHAVY, Petr; PROKOPOVA, Zdenka. Analysis and selection of a regression model for the Use Case Points method using a stepwise approach. Journal of Systems and Software, 2017, 125: 1-14.

[28] SILHAVY, Radek; SILHAVY, Petr; PROKOPOVA, Zdenka. Algorithmic optimisation method for improving Use Case Points estimation. PloS one, 2015, 10.11: e0141887.

[29] VORÁČOVÁ, Šárka. Regresní a korelační analýza [online]. Dostupné také z: https://www.fd.cvut.cz/department/k611/pedagog/K611THO_soubory/0_regrese.pdf

[30] Akaikovo informační kriterium [online]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-genomickych-a-proteomickych-dat--analyza-sekvenci-dna--substitucni-model--vyber-substitucniho-modelu--akaikovo-informacni-kriterium

[31] *Mean Absolute Percent Error (MAPE)* [online]. Dostupné také z: http://www.van-guardsw.com/business-forecasting-101/mean-absolute-percent-error/

[32] NAGPAL, Geeta, Moin UDDIN a Arvinder KAUR. *ESTIMATING PROJECT DEVELOPMENT EFFORT USING CLUSTERED REGRESSION APPROACH* [online]. Jalandhar. Dostupné také z: http://airccj.org/CSCP/vol3/csit3654.pdf

[33] *Shlukování* [online]. České vysoké učení technické v Praze. Dostupné také z: https://cw.fel.cvut.cz/wiki/_media/courses/a6m33dvz/dvz2014-02-shlukovani.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AIC	Akaike Information Criterion
AICc	Akaike Information Criterion correction
API	Application Programming Interface
APM	Application Point model
BIC	Bayesian Information Criterion
COCOMO	Constructive Cost Model
ECF	Environmental Complexity Factor
EDM	Early Design model
FPA	Function Points Analysis
GUI	Graphic User Interface
KLOC	Kilo Lines of code
MAPE	Mean Absolute Percentage Error
MRE	Magnitude of Relative Error
MSE	Mean Squared Error
PAM	Post-Architecture model
PF	Productivity Factor
RAD	Rapid Application Development
RMSE	Root Mean Squared Error
SSE	Sum of Squared Errors
SSR	Sum of Squares Regression
SST	Sum of Squares Total
TCF	Technical Complexity Factor
UAW	Unadjusted Actor Weights
UCP	Use Case Points

UUCW Unadjusted Use Case Weights

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SEZNAM PŘÍLOH NA CD

Obsah disku:

- Konečná verze diplomové práce
- Regresní modely modelované v programu Matlab
- Dataset hodnot