Využití analytického programování pro odhady časových náročností vývoje softwarových projektů

Ing. Tomáš Urbánek, Ph.D.

Teze disertační práce

Univerzita Tomáše Bati ve Zlíně

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Teze dizertační práce

Využití analytického programování pro odhady časových náročností vývoje softwarových projektů

Usage of analytical programming for effort estimation of software projects

Autor: **Ing. Tomáš Urbánek, Ph.D.**

Studijní program: Inženýrská informatika P3902
Studijní obor: Inženýrská informatika 3902V023

Školitel: doc. Ing. Zdenka Prokopová, CSc.

Oponenti: prof. RNDr. Ing. Miloš Šeda, Ph.D.
doc. RNDr. PaedDr. Hashim Habiballa, PhD, Ph.D.
doc. Ing. Petr Čermák, Ph.D.

Zlín, Prosinec 2020

Vydala **Univerzita Tomáše Bati ve Zlíně** v edici **Doctoral Thesis Summary** v roce 2020.

Klíčová slova: *Odhad časové náročnosti, Softwarové inženýrství, Odhadování, Use Case Points*

Key words: *Effort estimations, Software engineering, Predictions, Use Case Points*

Plná verze dizertační práce je dostupná v Knihovně UTB ve Zlíně.

ISBN 978-80-7454-975-5

ABSTRAKT

Disertační práce je zaměřena na výzkum v oblasti softwarového inženýrství specificky na získávání odhadů časového úsilí. Tato práce jako celek má přispět k vytvoření frameworku pro přesnější odhad tohoto úsilí pomocí metod symbolické regrese. Byly řešeny otázky linearity mezi odhadem a skutečným časovým úsilím, dále možnosti optimalizace frameworku, jak pomocí snížení velikosti prohledávaného prostoru, tak změnou účelové funkce. Dále je v práci porovnán odhad pomocí nového frameworku s odhady vypočítanými dalšími zkoumanými algoritmy. Framework byl testován na dvou datasetech s křížovou validací. Na datasetu s menším počtem vzorků dosahuje prezentovaný framework průměrné relativní chyby 40 %, což je zpřesnění oproti ostatním použitým metodám v průměru až 20 % a oproti standardní UCP rovnici až o 18 %. Na větším datasetu se průměrná relativní chyba pohybuje na hodnotě 8 %, která je srovnatelná s ostatními použitými metodami a taktéž zpřesněním oproti UCP rovnici až o 18 %.

ABSTRACT

This thesis is focused on research in the field of software engineering; specifically on obtaining estimates of time effort. This work as a whole is intended to contribute to the creation of a framework for more accurate effort estimation. Presented effort estimation framework using symbolic regression methods. The issues of linearity between the estimate and the actual time effort were being undertaken, as well as the possibilities of optimizing the framework. Furthermore, this thesis compares the estimates using a new framework with estimates providing other models used in this field. This new framework was tested on two datasets with a cross-validation technique. On a dataset with a smaller number of samples, the presented framework achieves an average relative error of 40 %. This result is up to 20 % on average more accurate compared to other methods. And also up to 18 % more accurate compared to the standard UCP equation. On a larger dataset, the average relative error is around 8 %, which is comparable to the other methods used. Moreover, these results on a larger dataset are also a refinement of up to 18 % compared to the UCP equation.

OBSAH

1 ÚVOD

Softwarová řešení jsou nedílnou součástí ekonomického růstu téměř ve všech průmyslových odvětvích. Má však také velké sociální dopady, jak dokládá studie Beckert et al. [2]. Většina dnešního zboží a služeb je částečně nebo z většiny realizována formou softwarového systému. Z tohoto důvodu naše závislost na softwaru neustále roste. Mnoho produktů, které byly tradičně realizovány formou hardwaru, jsou dnes řešeny formou softwaru. Zároveň velikost a komplexita těchto softwarových systému rychle narůstá a software nyní musí splňovat mnoho těžce splnitelných podmínek. Musí být rychlý, více inteligentní, musí snižovat své hardwarové nároky, musí být lehce udržovatelný atd.

Vývoj softwaru je též velmi závislý na lidských zdrojích. Většina práce na softwaru je intelektuální a do jisté míry kreativní lidská činnost, což přináší do vývoje softwaru mnoho neurčitostí. Tradičně je odhad časového úsilí využíván pro plánování a sledování zdrojů, například množstvím vývojářů potřebných pro dokončení softwarového projektu, a podobně. Proto rostou na důležitosti metody pro odhadování časového úsilí. Odhad časového úsilí v softwarovém inženýrství je definován jako úsilí nutné k dokončení softwarového projektu [11]. A je jednou ze základních částí tak zvané podpory softwarového procesu.

Odhad úsilí by měl sloužit jako opora pro řízení a správné rozhodování, proto je nutné poskytnout projektovým manažerům přesné odhady tohoto úsilí. Přesný odhad je takový, který poskytne projektovému manažerovi oporu a vedení k dosažení úspěšného projektového řízení a úspěšného dokončení projektu [24]. Přesné odhady pomáhají vytvořit plán, podle kterého se bude vývojový cyklus řídit. Odhad časového úsilí nebude nikdy úplně přesný a v konečném důsledku může být buď podhodnocen nebo nadhodnocen. Podhodnocení vzniká v případě, kdy reálná hodnota je vyšší než náš odhad. Při podhodnocených odhadech časových náročností pak může docházet k nedodržování termínů vývoje a psychického tlaku na vývojový tým. Naproti tomu nadhodnocení vzniká v případě, kdy reálná hodnota je nižší než náš odhad. V tomto případě softwarová společnost nadhodnotila cenu za vývoj nového softwarového produktu a došlo tak k překročení rozpočtu.

2 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Výzkum Burgess et al. [6] používá pro odhady velmi podobnou techniku jako v této práci. Autoři používají pro symbolickou regresi algoritmus Genetického programování [13] a pro odhad základních parametrů projektu je použita metoda COCOMO. Závěry ohledně použitelnosti těchto metod potvrdila také studie Sheta et al. [19], kde využívají také techniky Genetického programování. Byly generovány matematické popisy pomocí genetického programování a tyto byly porovnávány s modely v dostupné literatuře. Autoři studie tvrdí, že genetické programování dosáhlo lepších výsledků než ostatní modely. Nicméně problémy s nastavením resp. komplexností nastavení algoritmu genetického programování uvádí také studie Lefley et al. [15]. Autoři taktéž tvrdí, že genetické programování dosahovalo konzistentně lepších výsledků než použití metod umělých neuronových sítí a metod založených na nejbližším sousedovi.

Studie Shan et al. [18] využívá pro odhady časových náročností algoritmus Grammar Guided Genetic Programming. Jedná se o obdobu algoritmu genetického programování, kde je evoluce řízená gramatikou. Výsledky byly porovnávány s lineární a log. regresí. Průměrně bylo dosahováno snížení MMRE o 2,67. Autoři studie také uvádí, že důvodem pro použití metod symbolické regrese je, že výsledky jsou interpretovatelné oproti např. neuronovým sítím, kde je výsledkem černá skříňka [14].

Vzhledem k současnému stavu výzkumu v této oblasti se zdá použití symbolické regrese jako vhodná metoda pro návrh nového algoritmu. Hlavně z důvodů, že řešením je matematický popis, který je interpretovatelný v porovnání s výsledky např. neuronových sítí. Interpretovatelný model může být snadno zkontrolován, auditován, a tím roste důvěra v použitý model [8]. Většina citovaných studií se potýká s problémy nastavení algoritmu genetického programování. Z tohoto důvodu byl pro tuto práci použit algoritmus analytického programování [27], který nemá žádné nastavení. Nastavuje se pouze evoluční algoritmus pro vyhledávání. Dále se výše zmiňované studie potýkají s problematikou "bloat"tedy bobtnání řešení, který je velmi známým efektem u genetického programování [3]. Analytické programování tímto efektem netrpí.

3 CÍLE DIZERTAČNÍ PRÁCE

3.1 Hlavní cíl

- Návrh frameworku k získání přesnějších odhadů časových náročností softwarových projektů

3.2 Dílčí podpůrné cíle

- Provést rešerši poznatků domácí i světové odborné literatury týkající se odhadů časových náročností softwarových projektů
- Získání datových sad pro experimenty a simulační studie
- Omezení vlivu lidského faktoru na odhady časových náročností
- Kritické ověření možnosti použití lineárních modelů pro odhad časových náročností softwarových projektů

3.3 Stanovení výzkumných otázek

1. Existuje lineární závislost mezi ohodnocením softwarového projektu a skutečnou časovou náročností softwarového projektu?
2. Které parametry metody UCP mají největší vliv na přesnější odhad při použití nového frameworku?
3. Jaký je vliv různých účelových funkcí na přesnost odhadu a která účelová funkce poskytne nejpřesnější odhad?
4. Je odhad pomocí nového frameworku přesnější než odhad podle standardní UCP rovnice?
5. Je odhad pomocí nového frameworku přesnější než odhady jiných algoritmů: mnohonásobná lineární regrese, kroková lineární regrese, neuronová síť, ridge regrese, LASSO regrese, rozhodovací strom CART, obecný lineární model, k-nearest neighbors, support vector machine

4 TEORETICKÝ RÁMEC

Existuje mnoho metod pro odhadování časových náročností softwarových projektů. Z taxonomického hlediska můžeme metody pro odhadování rozdělit do tří hlavních kategorií [24].

Metody řízené daty odkazují na metody, které ke své predikci používají kvantitativní analýzu historických dat. Relace mezi řešeným projektem a jeho charakteristikou jsou vysvětlovány pomocí dat shromážděných z už dokončených projektů. Tyto relace jsou dále využity k odhadům nového projektu. Nejvýznamnější a také nejpoužívanější metody v této kategorii jsou COCOMO [4], FP [1] a metoda UCP [10].

Metody založené na expertním úsudku jsou nejpoužívanější skupinou pro odhadování úsilí. Tyto metody využívají konzultaci projektu s jedním či více experty, kteří mají zkušenosti s daným typem řešené problematiky [9]. Často se také využívá analogií současného projektu s projektem řešeným v minulosti. Nejznámější metody z této kategorie jsou Delphi, Wideband Delphi [4] a další.

Hybridní metody jsou založeny na kombinování metod z kategorie řízené daty a metod založených na expertním úsudku. Protože nepanuje shoda, které metody jsou přesnější, hybridní metody si berou silné stránky z nabízených metod a snaží se potlačit jejich negativní vlastnosti. Nejznámější metodou z této kategorie je metoda "Cost estimation, Benchmarking, and Risk Assessment"(dále CoBRA) [5] či metody založené na Bayesově teorému.

Taxonomie odhadovacích metod používaných v softwarovém inženýrství je podle různých autorů rozdílná a obecně lze říci, že mezi autory nepanuje shoda o konkrétní taxonomii metod použitých pro odhadování [12].

4.1 Metoda Use Case Points

Je metoda využívaná v softwarovém inženýrství pro odhadování časových náročností softwarových projektů. Je založena na konceptu ohodnocování tzv. případů užití. Vytváření případů užití je nedílnou a standardní součástí moderního modelování softwarových projektů pomocí jazyka UML. Metoda UCP byla poprvé prezentována Gustavem Karnerem v roce 1993 [10]. Projektový manažér musí odhadnout 4 parametry projektu. Tyto parametry jsou UUCW, UAW, TCF a ECF.

Neupravená váha případu užití UUCW

Neupravená váha případu užití se zabývá komplexitou vyvíjeného systému. Komplexita se vypočítá z případů užití. Metoda UCP používá tři kategorie pro klasifikaci neupravené váhy případu užití. Vliv jednotlivých kategorii se řeší sumací a váhováním.

Neupravená váha aktérů UAW

Je to parametr softwarového projektu zabývající se komplexitou aktérů vyvíjeného systému. Metoda UCP používá tři kategorie pro klasifikaci neupravené váhy aktérů. Vliv jednotlivých kategorií se řeší sumací a váhováním.

Technický faktor TCF

Metoda UCP ohodnocuje softwarový projekt třinácti technickými faktory. Tyto se zaměřují zejména na technickou komplexitu vyvíjeného systému. Hodnota síly vlivu technického faktoru je od 0 (faktor není důležitý) do 5 (faktor je velmi důležitý). Tato hodnota je následně násobena váhou daného faktoru a sečtena pro všechny faktory. Na rozdíl od předchozích parametrů, zde dochází ještě ke korekci výsledného vlivu parametru.

Faktor prostředí ECF

Metoda UCP obsahuje 8 faktorů prostředí. Tyto se zaměřují na komplexitu prostředí vyvíjeného systému. Hodnota síly vlivu faktoru prostředí je od 0 (faktor není důležitý) do 5 (faktor je velmi důležitý). Tato hodnota je následně násobena váhou daného faktoru a sečtena pro všechny faktory. U tohoto parametru taktéž dochází ke korekci výsledného vlivu parametru.

Faktor produktivity

Než je vypočten výsledný odhad časového úsilí metodou UCP je potřeba stanovit faktor produktivity (dále FP). Je to konstanta, která udává, kolik je potřeba člověkohodin na jeden bod případu užití. Faktor produktivity byl stanoven Gustavem Karnerem v roce 1993 [10] a jeho hodnota byla vypočtena na 20 hodin na jeden bod případu užití.

Výpočet odhadu

Pro kalkulaci počtu bodů případu užití je používána rovnice 4.1.

$$UCP = (UUCW + UAW) \times TCF \times ECF, \tag{4.1}$$

kde UCP je počet bodů případu užití.

Pro kalkulaci výsledného odhadu je používána rovnice 4.2.

$$EE = UCP \times FP, \tag{4.2}$$

kde EE je výsledný odhad v jednotkách člověkohodin, UCP je počet bodů případu užití a FP je faktor produktivity.

4.2 Symbolická regrese

Symbolická regrese, také někdy označována jako metoda evoluční syntézy struktur, je úloha identifikace matematického popisu z experimentálně získaných dat. Symbolická regrese je schopná syntetizovat jednotlivé matematické popisy i komplexní programy v určitém programovacím jazyce. Metody symbolické regrese nehledají pouze parametry regresní funkce, ale jsou schopny získat samotnou regresní funkci. Důležitými pojmy v této oblasti umělé inteligence jsou pojmy neterminální a terminální symboly.

Neterminální symboly jsou funkce, ať už matematické, či uživatelsky definované (realizované formou zdrojového kódu). Každá funkce, ať už matematická či programátorsky definovaná, má určitý počet parametrů. Tím se odlišuje od terminálního symbolu, který žádné parametry nemá. Z toho vyplývá, že do parametru neterminálního symbolu můžeme přiřadit další neterminální popř. terminální symbol. Tímto procesem lze vytvořit např. datovou strukturu stromu, kde neterminální symboly jsou uzly a terminální symboly jsou listy stromu. Terminální symboly jsou symboly konečné a nelze je nadále dělit. Terminální symboly jsou tedy parametry funkcí. Mohou to být proměnné, případně konstanty.

4.2.1 Analytické programování

Analytické programování lze chápat jako transformaci množiny dat, která bude předávána evolučnímu algoritmu, např. Self-Organizing Migrating Algorithm (dále SOMA) nebo diferenciální evoluci (dále DE), k ohodnocení a řízení další evoluce. Analytické programování, na rozdíl od jiných metod evoluční syntézy struktur, nevyužívá žádné reprezentace dat typu strom nebo gramatiky. Stejně jako v ostatních algoritmech evoluční syntézy struktur se používají terminální a neterminální symboly. Všechny použité funkce se setřídí podle počtu jejich parametrů. To znamená, že např. funkce $Sin(\cdot)$ obsahuje jeden parametr, funkce + obsahuje dva parametry, v tomto případě můžeme funkci + chápat jako $plus(\cdot, \cdot)$. Tímto získáme množiny funkcí, označované jako general function set (dále GFS), kde funkce s vyšším počtem parametrů jsou nadmnožinou funkcí s nižším počtem parametrů. Terminální symboly jako proměnné a konstanty jsou podmnožinou všech ostatních funkcí, jelikož nemají žádný parametr. Proces evoluce využívá schopnosti evolučních algoritmů pracovat s diskrétními hodnotami. Každý jedinec se skládá z celočíselných hodnot, které jsou ukazatelem do tabulky terminálních a neterminálních symbolů. Jakmile je jedinec sestaven, je ohodnocen účelovou funkcí některého z použitých evolučních algoritmů, např. SOMA nebo DE. [28, 25].

4.3 Diferenciální evoluce

Diferenciální evoluce vznikla v roce 1995 a jejími autory jsou Ken Price a Rainer Storm [22]. Diferenciální evoluce sdílí určité aspekty s genetickými algoritmy, jako je např. tvorba potomků, generace, evoluce apod. Algoritmus diferenciální evoluce byl vyvinut z genetického žíhání úpravou mutace, která se nazývá diferenciální mutace. První verze diferenciální evoluce nedostačovaly pro použití na široké množině optimalizačních úloh. Až třetí verze algoritmu diferenciální evoluce byla přijata s kladným ohlasem. Algoritmus diferenciální evoluce se řídí čtyřmi parametry NP, F, CR a generations.

Jako u všech evolučních algoritmů je nejprve vytvořena prvotní generace. Počet vytvořených jedinců určuje parametr NP. Pro evoluční proces jsou vybráni čtyři jedinci. Jedinci vstupují do evolučního procesu mutací. Mutace probíhá ze tří náhodně vybraných jedinců. Mutace vytváří tzv. šumový vektor. Tento

vektor je tvořen pomocí vzorce 4.3.

$$v_j = x^G_{r3,j} + F \cdot \left(x^G_{r_1,j} - x^G_{r2,j}\right), \tag{4.3}$$

kde v_j je šumový vektor, $x^G_{r3,j}$ je vektor reprezentující třetího jedince, $x^G_{r_1,j}$ je vektor reprezentující 1. jedince, $x^G_{r2,j}$ je vektor reprezentující druhého jedince, F reprezentuje parametr mutační konstanty, G reprezentuje generaci. Rozdíl dvou náhodně vybraných jedinců je vynásoben mutační konstantou F, a tento výsledek je přičten k třetímu, zatím nepoužitému jedinci. Při křížení hraje důležitou roli právě šumový vektor a čtvrtý zatím nepoužitý jedinec. Pro každou položku vektoru se vygeneruje náhodné číslo v intervalu $< 0; 1 >$. Toto číslo se porovná s parametrem CR. Jestliže je číslo menší než parametr CR, do nového vektoru, tzv. zkušebního vektoru, se vybere položka ze šumového vektoru. Jestliže je vygenerované číslo naopak větší, do šumového vektoru se vybere položka ze čtvrtého jedince. Jedinci dále vstupují do procesu ohodnocení účelovou funkcí. Každému jedinci je určena jeho vhodnost. Po určení vhodnosti všech jedinců jsou vybráni jedinci, kteří vstoupí jako rodiče do nové populace. Tato populace opět projde procesy mutací, křížení, ohodnocení a výběrem do nové populace, dokud není algoritmus zastaven počtem iterací, který udává parametr Generations nebo jinou ukončovací podmínkou [25].

4.4 Kritéria kvality

Mnoho vědeckých studií se zabývá otázkou jaké zvolit kritérium pro kvalitní řešení. Vzhledem k velkému počtu různých kritérií kvality budou uvedeny pouze nejpoužívanější.

4.4.1 Kritérium kvality MMRE

Hodnoty této funkce jsou vždy kladná čísla a hodnoty blížící se nule jsou známkou kvalitnějšího řešení. Mezi její hlavní výhody patří porovnatelnost kvality řešení mezi různými metodami. Toto tvrzení vychází z rovnice pro MMRE, kde jsou odchylky od reálné hodnoty děleny právě velikostí reálné hodnoty.

$$MMRE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i}, \tag{4.4}$$

kde n je celkový počet vzorků, y_i je naměřená hodnota a $\hat{y}_i$ odhadovaná hodnota.

4.4.2 Kritérium kvality LAD

Jedná se o sumu odchylek od skutečné hodnoty. Hodnoty této funkce jsou vždy kladná čísla a zároveň hodnoty blížící se nule jsou známkou kvalitnějšího řešení. Je využíváno zejména robustnosti daného kritéria díky vlastnostem absolutní hodnoty obsažené v této funkci. Nevýhodou pak je, že nemusí existovat pouze jedno optimální řešení.

$$LAD = \sum_{i=1}^{n} |y_i - \hat{y}_i|, \tag{4.5}$$

kde n je celkový počet vzorků, y_i je naměřená hodnota a $\hat{y}_i$ odhadovaná hodnota.

4.4.3 Kritérium kvality MSE

Kritérium se využívá zejména ve statistice a datových vědách, kde je často používáno k ohodnocování statistických modelů. Hodnoty této funkce jsou vždy kladná čísla a hodnoty blížící se nule jsou známkou kvalitnějšího řešení. V této práci je MSE využita jako účelová funkce i jako evaluace kvality daného řešení. Minimalizací kritéria MSE dochází k minimalizaci rozptylu daného modelu.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{4.6}$$

kde n je celkový počet vzorků, y_i je naměřená hodnota a $\hat{y}_i$ odhadovaná hodnota.

5 ZVOLENÉ METODY ZPRACOVÁNÍ

5.1 Metoda analýzy

Metoda analýzy je proces dekompozice celku (jevu, předmětu) na části. Dochází k rozboru vlastností, zkoumání vztahů a faktů od celku k částem. Analýza je založena na předpokladu, že každý zkoumaný systém lze rozložit na množiny prvků, které jsou spojeny vlastnostmi a jednotlivými vazbami. Analýza umožňuje oddělit podstatné od nepodstatného a odlišit trvalé vztahy od vztahů nahodilých [7]. Tato metoda bude použita při zkoumání výsledných matematických modelů. Na základě zkoumání a analýzy těchto konkrétních modelů budou vymezeny všeobecné požadavky na následující generování nových modelů.

5.2 Metoda experimentu

Jedná se o empirickou metodu, která je zaměřena na testování a ověření vytvořených hypotéz za stanovených podmínek. Cílem je potvrdit nebo vyvrátit platnost stanovených hypotéz [26]. Metoda experimentu je jedna z nejdůležitějších metod nutných pro naplňování cílů disertační práce. V disertační práci bude metoda experimentu použita ve fázi ověřování funkčnosti navrhovaného frameworku.

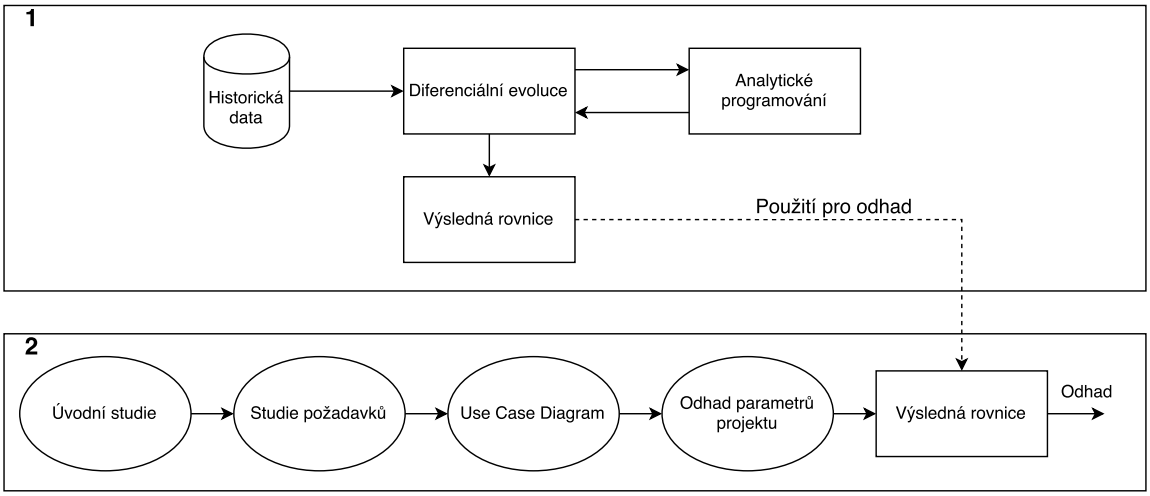
5.3 Metody zpracování disertační práce

V první řadě je nutné stanovit výzkumné otázky. Z výzkumných otázek budou dále formulovány konkrétní hypotézy, které budou podrobeny klasickým metodám matematické statistiky jako je deskriptivní (popisná) statistika a inferenční statistika (testování statistických hypotéz a vytváření intervalů spolehlivosti). Dále je třeba určit průběh experimentů a následný sběr dat. Poté mohou být použity statistické metody a data mohou být vyhodnocena.

6 HLAVNÍ VÝSLEDKY PRÁCE

6.1 Popis navrženého frameworku odhadování

Po konstrukci diagramu případu užití, který je standardní součástí projektové dokumentace, dochází k ohodnocení složitosti jednotlivých případů užití. Pro výpočet pomocí metody UCP je dále potřeba odhadnout ostatní charakteristiky projektu, jako je např. složitost jednotlivých aktérů a technické faktory. Po těchto úkonech máme k dispozici čtyři základní charakteristiky softwarového projektu, jak je popsáno v kapitole 4.1. V této práci je předpokládáno, že rovnice 4.1 a 4.2 jsou zastaralé a neodráží požadavky na vývoj dnešních softwarových systémů, proto je zde snaha tyto rovnice nahradit jiným matematickým modelem, který by dokázal odhad zpřesnit. Jako algoritmus pro syntézu rovnic je použit právě algoritmus analytického programování.



Obr. 6.1 Diagram funkce navrhovaného frameworku

Princip funkce navrhovaného frameworku je prezentován na obrázku 6.1. Návrh je rozdělen na dva bloky.

Blok 1 zobrazuje hlavní funkci vytvoření nové rovnice pro odhad. Softwarová společnost vlastní historická data vyvíjených softwarových produktů v databázovém systému. Zaznamenávané údaje jsou parametry metody UCP tzn. UUCW, UAW, ECF, TCF a dále skutečná časová náročnost softwarového projektu. Tato data jsou zpracována algoritmem diferenciální evoluce a analytického programování. Výsledkem tohoto procesu je nový matematický model, který reflektuje informace z historických dat.

Blok 2 zobrazuje odhad nového softwarového projektu. Jakmile má softwarový inženýr ohodnocen softwarový projekt podle metodiky založené na UCP, použije rovnici připravenou výše zmíněným algoritmem. Připravený model by měl být přesnější vzhledem k využití historických dat. Jakmile softwarová společnost dokončí projekt, je tento považován za historický (nese novou informaci), je uložen do databázového systému a může být použit k dalšímu zpřesňování odhadů.

6.2 Popis získaných dat a proměnných

Data byla získána především z publikací a výzkumu autorů věnujících se podobné problematice odhadování časových náročností softwarových produktů.

- Dataset označen D1 je sloučením dvou datasetů :
  - Dataset z Technické univerzity v Poznani [16]
  - Dataset nacházející se v článku autora Subriady et al. [23]
- Dataset označen D2 získaný od autorů Šilhavý et al. [20, 21]

Dataset získaný z technické univerzity v Poznani od autora publikujícího v oboru softwarového inženýrství Ochodek et al. [16] sestává ze 14 softwarových projektů. Tyto softwarové projekty byly programovány v různých programovacích jazycích. Převládá zde však programovací jazyk JAVA.

Dataset získaný z článku autora Subriady et al. [23] sestává z 10 softwarových projektů. Tyto softwarové projekty byly programovány v programovacích jazycích určených pro webové programování, kde převládá programovací jazyk PHP. Co se týká rozsahu jednotlivých softwarových projektů, jedná se především o webové aplikace. Tyto dva výše zmíněné datasety byly sloučeny pod označením D1 a obsahují 24 softwarových projektů.

Dataset získaný od autorů Šilhavý et al. [20, 21] obsahuje 70 pozorování. Tyto softwarové projekty byly programovány v programovacích jazycích určených zejména pro desktopové aplikace, kde převládá programovací jazyk C# a JAVA. Tento dataset bude dále v textu označován jako D2.

Tab. 6.1 Výpočet průměru a relativní četnosti jednotlivých znaků

Dataset	R^2	Shapiro-Wilk	Breusch-Pagan
D1	0,66	0,92	1,00
D2	0,33	0,82	0,68

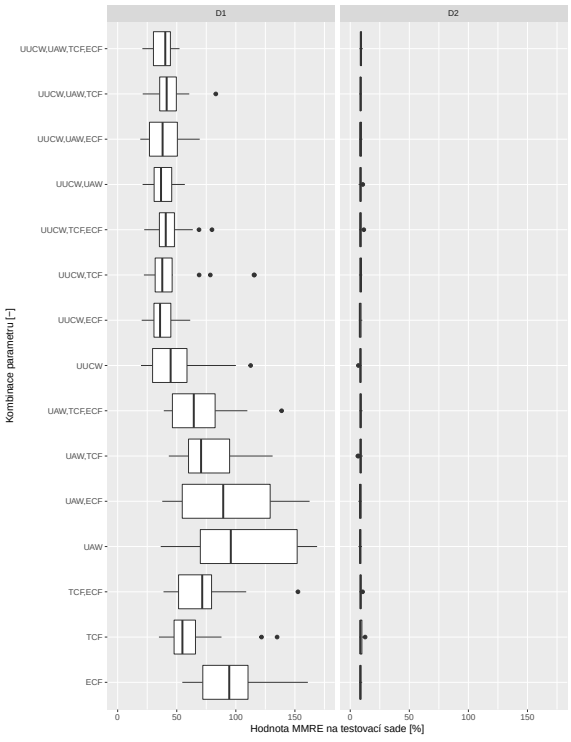
6.3 Výzkumná Otázka 1

V tabulce 6.1 je zaznamenán výpočet indexu determinace, Shapiro-Wilk testu a Breusch-Paganova testu. Je nutné si povšimnout zejména hodnot R^2, které se mezi datasety významně liší. Velmi mnoho průběhů obsahovalo normálně rozdělená rezidua, u D1 je to 92 % cyklů u D2 je to 82 % cyklů. U obou datasetů z těchto 27 cyklů všechny průchody nezaznamenaly splnění homoskedasticity reziduí.

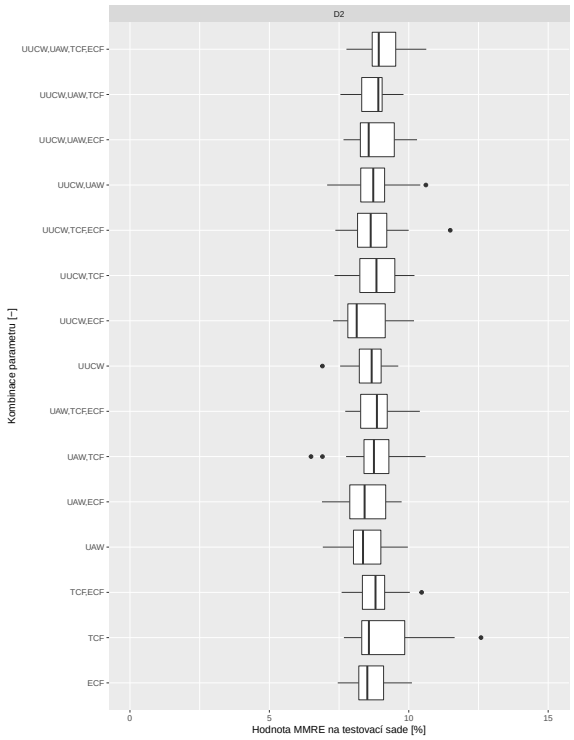
Hypotéza o nelineární závislosti mezi odhadem parametrů softwarového projektu a skutečnou časovou náročností nebyla ze získaných dat potvrzena. A tedy je možné apriori předpokládat lineární závislost. Nicméně, a na to je třeba upozornit, před aplikací mnohonásobné lineární regrese je nutné zkontrolovat podmínky pro její použití. Důvodem je zejména to, že v datasetu D2 byl Breusch-Pagan test z frekvencí 23% zamítnut. Velmi diskutabilní jsou taky výsledky vysvětlené variability, kde zejména u datasetu D2 byla naměřena velmi nízká hodnota koeficientu determinace 0,3 a u datasetu D1 byla naměřena hraniční hodnota blížící se 0,6. Použití mnohonásobné lineární regrese se z výše uvedených důvodů nejeví jako vhodná metoda pro tento typ dat.

6.4 Výzkumná Otázka 2

Výsledky tohoto experimentu jsou znázorněny na grafu 6.2. Na datasetu D1 si můžeme povšimnout dvou důležitých zjištění. První rovnice, které měly přístup k parametru UUCW si v konečném hodnocení vedly lépe, zde bylo dosahováno relativní chyby kolem 40 %. Rovnice, které neměly k parametru UUCW přístup dosahovaly relativní chyby v průměru 80 %. Nutno podotknout, že v testovací sadě pro dataset D1 se nachází 33 % datasetu, tedy 8 softwarových projektů. Druhé zjištění bylo, že v porovnání s datasetem D2 jsou výsledky horší o



Obr. 6.2 Statistické porovnání jednotlivých skupin parametrů projektů datasetů D1 a D2



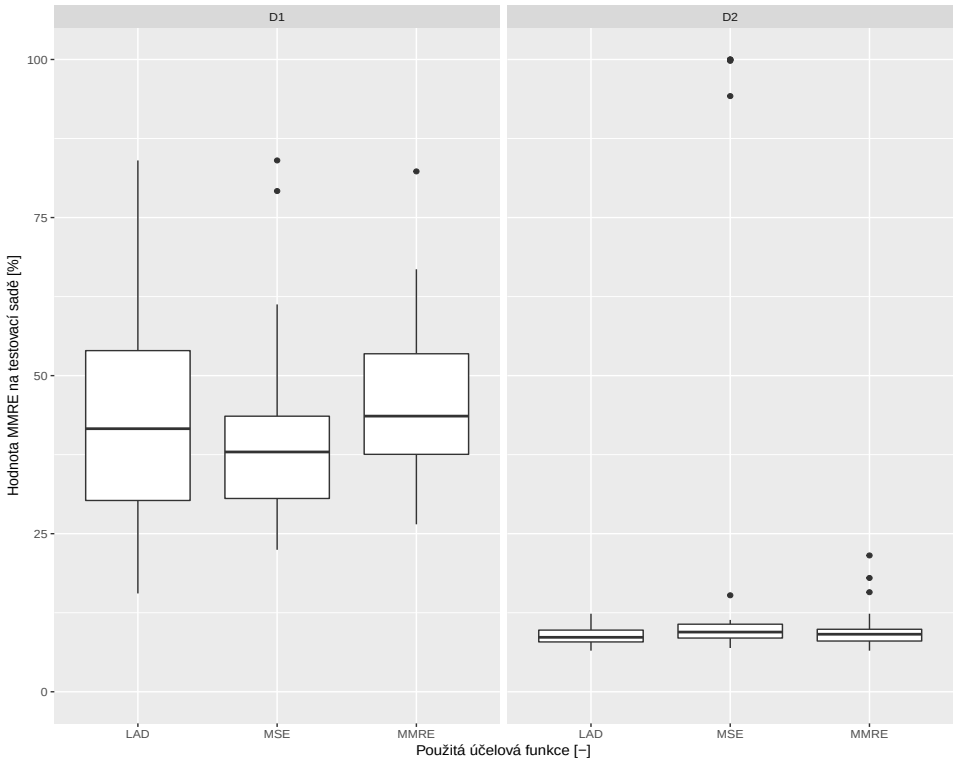
Obr. 6.3 Statistické porovnání jednotlivých skupin parametrů projektů datasetu D2

cca 30 % MMRE. Co se týká datasetu D2, zde si můžeme povšimnout, že všechny kombinace parametrů si vedou velmi podobně. Průměrná relativní chyba na testovacích datech datasetu D2 je 8,7 %. U datasetu D2 je menší rozptyl výsledných hodnot MMRE, než tomu bylo u datasetu D1. Průměr MMRE se pohybuje kolem hodnoty 8 % s minimem 6,9 % MMRE. Je zde také vidět, že žádná s kombinací parametrů nemá na výslednou hodnotu MMRE zásadní vliv.

Tab. 6.2 Výpočet hypotéz ANOVA pro oba datasety

Dataset	Statistika	Stupňů volnosti	p-hodnota
D1	$F = 17,78$	14	$< 0{,}01$
D2	$F = 1,162$	14	0,304

V tabulce 6.2 je vidět, že p-hodnota pro dataset D1 je pod hodnotou 0,05. A tedy máme dostatek důkazů pro zamítnutí nulové hypotézy o shodném průměru všech skupin. Dále tedy budeme předpokládat, že mezi skupinami jsou statisticky významné rozdíly v průměru dosahovaných hodnot MMRE. U da-



Obr. 6.4 Statistické porovnání jednotlivých skupin použitých účelových funkcí

tasetu D2 můžeme v tabulce vidět, že p-hodnota je větší než hraniční hodnota 0,05. Zde nezamítáme nulovou hypotézu a dále v textu budeme předpokládat, že mezi skupinami není statisticky významný rozdíl.

Hlavní motivací, řešení této výzkumné otázky je fakt, že prohledávání prostoru o takové dimensionalitě je výpočetně velmi náročné. Pro menší dataset D1 je zjištění z grafu vypovídající. Jestliže bylo povoleno použití parametru UUCW, tak framework dosahoval statisticky významně lepších výsledků, než kdyby tam parametr UUCW nebyl. Lze tedy usoudit, že aby nový framework vygeneroval přesnější rovnici, je potřeba, aby byl použit parametr UUCW. Pro větší dataset D2 není žádný z parametrů klíčový ve smyslu zvýšení zpřesnění, zde měly všechny parametry stejnou důležitost. Z výsledku statistického šetření je patrné, že mezi průměry relativních chyb není možné pozorovat statisticky významný rozdíl, jak bylo dříve dokázáno testem ANOVA.

6.5 Výzkumná Otázka 3

Výsledky tohoto experimentu jsou znázorněny na grafu 6.4. Na datasetu D1 si můžeme povšimnout, že všechny 3 použité účelové funkce mají podobný

medián a navzájem se překrývají. Dále je zajímavé si povšimnout, že funkce MSE vygenerovala nejvíce extrémních hodnot na grafu znázorněny tečkou. Na datasetu D1 dosahoval framework horších výsledků než na datasetu D2. Na datasetu D2 je nutné si povšimnout zejména účelové funkce MSE, která dosahovala výrazně horších výsledků než použití funkcí LAD a MMRE, z důvodů vygenerování více extrémních hodnot. Účelové funkce MMRE a LAD si vedla konzistentně podobně na obou datasetech.

Tab. 6.3 Výpočet hypotéz testu Kruskal-Wallis pro oba datasety

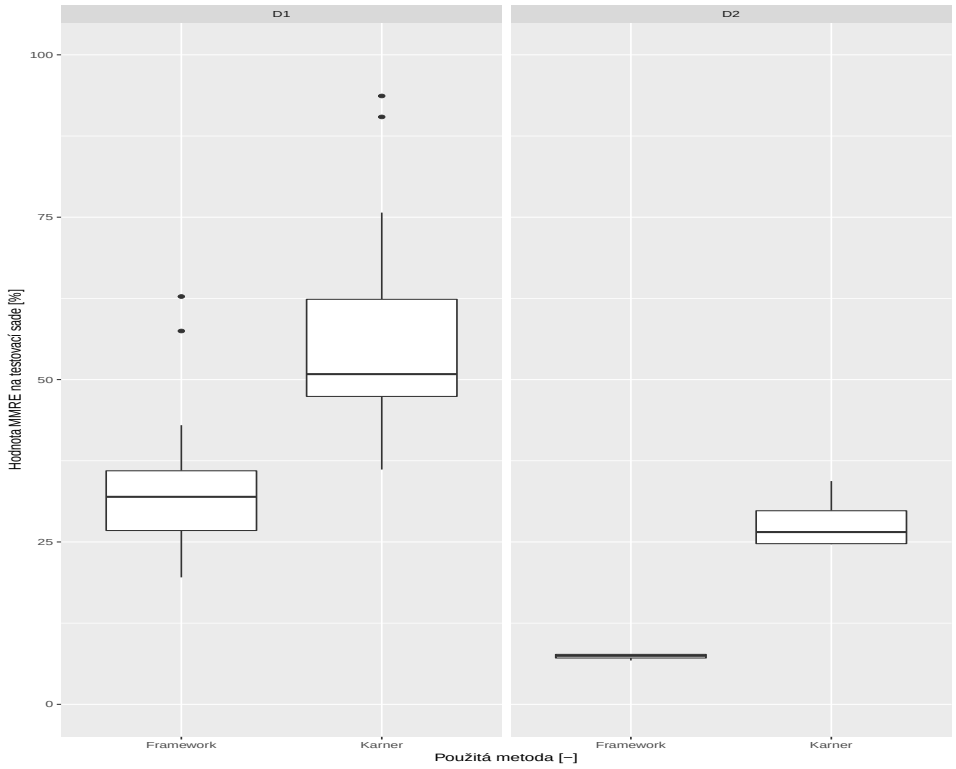
Dataset	Statistika	Stupňů volnosti	p-hodnota
D1	$\chi^2 = 7,9$	2	0,0192
D2	$\chi^2 = 8,21$	2	0,0165

V tabulce 6.3 je vidět, že p-hodnota pro dataset D1 je pod hraniční hodnotou 0,05. A tedy máme dostatek důkazů pro zamítnutí nulové hypotézy o shodném mediánu všech tří účelových funkcí. Dále tedy budeme předpokládat, že mezi skupinami jsou statisticky významné rozdíly v mediánu dosahovaných hodnot. U datasetu D2 jsme pod hraniční hodnotou pro zamítnutí nulové hypotézy na hladině významnosti 5 %. Máme tedy dostatek důkazů pro zamítnutí nulové hypotézy o shodnosti mediánů. Dále budeme předpokládat, že výsledek MMRE je statisticky významně rozdílný pro některou z použitých účelových funkcí.

Hlavní motivací, řešení této výzkumné otázky je fakt, že použití rozdílné účelové funkce může mít dopad na přesnost celého frameworku. Cílem tedy bylo dokázat, že je možné nalézt funkci, která pro tento druh úkolu není vhodná. Použití účelové funkce MSE se tedy z výsledků jeví jako nevhodné vzhledem k variabilitě a extrémním hodnotám. Dále v práci bude použita hlavně účelová funkce LAD, která má zejména na datasetu D2 nižší variabilitu a dosahuje hlubších minim.

6.6 Výzkumná Otázka 4

Výsledky tohoto experimentu jsou znázorněny na grafu 6.5. Pro dataset D1 je vygenerováno 12 vzorků pro dataset D2 jsou vygenerovány 2 vzorky. Jak můžeme sledovat pro oba datasety dosahuje framework nižších hodnot relativní



Obr. 6.5 Statistické porovnání prezentovaného frameworku a výpočtu pomocí Karnerovy rovnice

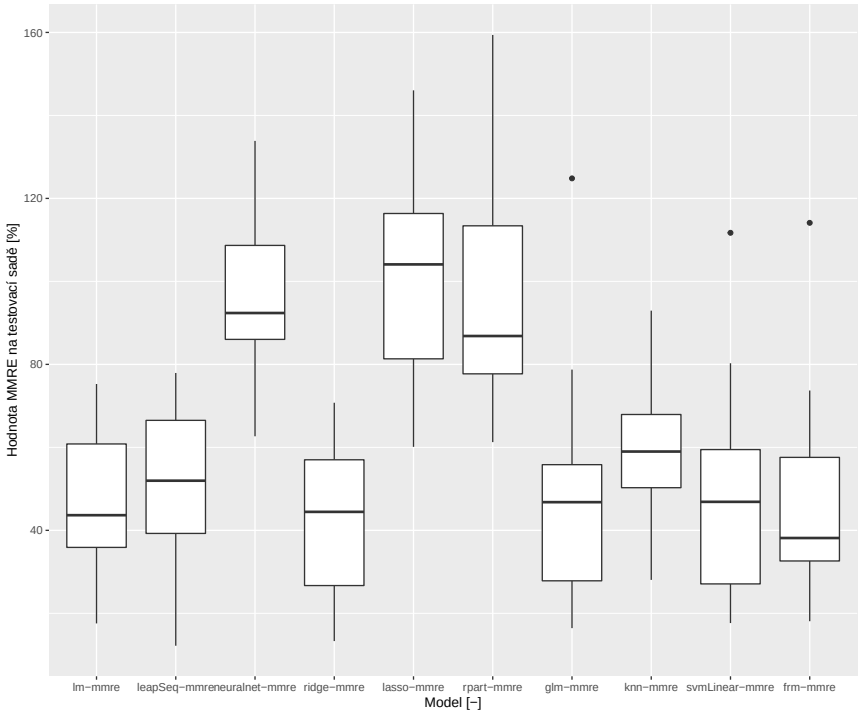
chyby MMRE. Větších rozdílů v přesnosti bylo dosahováno u datasetu D2, kde bylo sledováno zpřesnění oproti Karnerově rovnici cca. 19 %. Zajímavý je také fakt, že i Karnerova rovnice si ve druhém datasetu D2 vedla lépe než v datasetu D1. To je dáno pravděpodobně velikostí obou datasetů.

Tab. 6.4 Výpočet hypotéz jednostranného párového t-test pro oba datasety

Dataset	Statistika	Stupňů volnosti	p-hodnota
D1	$t = -9,69$	11	$< 0{,}01$
D2	$t = -9,23$	3	0,001342

V tabulce 6.4 je vidět, že p-hodnota pro oba datasety je menší než hraniční hodnota 0,05. Pro oba datasety tedy máme dostatek důkazu pro zamítnutí nulové hypotézy o shodnosti dosahovaných průměrných hodnot MMRE. Dále budeme předpokládat, že hodnoty dosahované prezentovaným frameworkem jsou statisticky významně nižší než pro standardní Karnerovu rovnici. Toto platí pro oba datasety tedy D1 i D2.

Výsledky dosažené při vyšetřování této výzkumné otázky jsou pro prezen-



Obr. 6.6 Statistické porovnání prezentovaného frameworku a ostatních modelů na datasetu D1

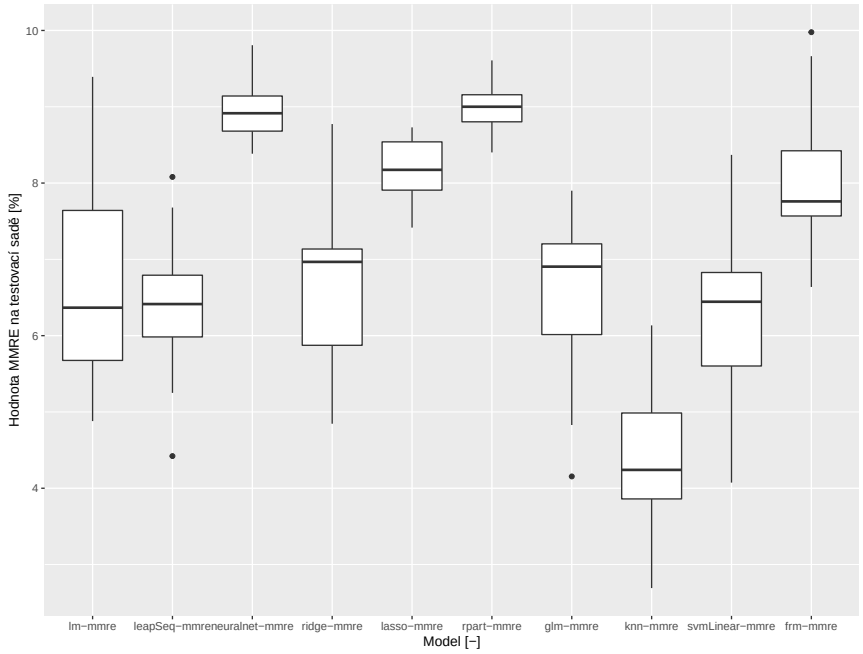
tovaný framework jedny z nejdůležitějších. Bylo statisticky ukázáno, že nový framework poskytuje konzistentně přesnější odhad na obou použitých datasetech. Průměrné zpřesnění se pohybuje na úrovni 18 % MMRE. Karnerova rovnice byla vypočtena s faktorem produktivity 20 [10]. Můžeme tedy říci, že průměrné zpřesnění oproti Karnerově rovnici se v nejlepším případě pohybuje na vypočtených 18 % MMRE.

6.7 Výzkumná Otázka 5

Na grafu 6.6 je patrné, že prezentovaný framework má nejnižší medián. Nicméně tento rozdíl nebude pravděpodobně statisticky významný oproti mnoha jiným modelům. Na grafu si můžeme dále povšimnout velmi špatné výkonnosti 3 modelů a těmi jsou neuronová síť, LASSO regrese a rozhodovací strom.

Na grafu 6.7 je patrné, že prezentovaný framework má na větším datasetu podstatně horší výsledky. Stále však dosahuje lepších výsledků než neurová síť, LASSO regrese a rozhodovací strom. Nejlepších výsledků dosahuje překvapivě algoritmus k-nearest neighbors.

V tabulce 6.5 je vidět, že p-hodnota pro oba datasety je pod hodnotou



Obr. 6.7 Statistické porovnání prezentovaného frameworku a ostatních modelů na datasetu D2

Tab. 6.5 Výpočet hypotéz ANOVA pro oba datasety

Dataset	Statistika	Stupňů volnosti	p-hodnota
D1	$F = 36,4$	9	$< 0{,}01$
D2	$F = 87,03$	9	$< 0{,}01$

0,05. A tedy máme dostatek důkazů pro zamítnutí nulové hypotézy o shodném průměru všech modelů. Dále tedy budeme předpokládat, že mezi skupinami jsou statisticky významné rozdíly v průměru dosahovaných hodnot MMRE.

Zde je patrné, že na menším datasetu poskytuje framework lepší výsledky než standardně používané modely. Velmi překvapivé jsou výsledky algoritmu k-nearest neighbors na datasetu D2. Autora to nutí k myšlence, že na trénovacích datech je velmi málo stupňů volnosti a celý prostor je možné rozdělit na několik klastrů. Zde se ukazuje nevýhoda symbolické regrese za použití pouze matematických funkcí, které nemají možnost rozhodování. Jestliže je daná domněnka správná, pak nový framework nemohl nalézt hlubší minimum a musel skončit v průměru mezi jednotlivými klastry. Na tento problém by mohlo pomoct definovaní konstruktu "if-else"do GFS. Nicméně tato práce se věnuje zejména matematickým funkcím, proto tyto programátorské konstrukty nebyly do GFS analytického programování zavedeny.

7 PŘÍNOS PRÁCE PRO VĚDU A PRAXI

7.1 Přínos pro vědu

Přínosem pro vědu je především framework vzniklý při tomto výzkumu. Byla ověřena funkce nového frameworku na odhadování časových náročností softwarových projektů. Oproti běžně používané Karnerově rovnici se průměrné zpřesnění metody pohybuje na úrovni až 18 %. Přínosem je tedy funkční framework a dále mnohé poznatky, které byly publikovány na konferencích a v odborných časopisech. Mezi tyto patří například testy vhodných účelových funkcí. Pro tento typ problému jsou to zejména statistické metriky LAD a MMRE, testy nastavení parametrů, jak analytického programování, tak diferenciální evoluce. Důležitým, ne však hlavním přínosem, je také zdokonalení algoritmu analytického programování. Tento je v současné době zkoumán a analyzován, viz práce [17]. Vylepšená metoda analytického programování by podle dosavadních testů měla vést k rychlejšímu generování funkcí. Přínosem pro vědu je určitě i komparativní studie uvedená ve výzkumné otázce 5, kde je porovnána přesnost různých algoritmů s novým frameworkem.

7.2 Přínos pro praxi

Získání přesnějších odhadů časových náročností povede k možnosti přesněji řídit softwarový cyklus. Přesnější odhady také umožní správně nacenit softwarový projekt, což je výhodné jak pro softwarovou společnost, tak pro zákazníka poptávajícího softwarový produkt. Tento nový framework by měl zpřesnit a zjednodušit odhady softwarových inženýrů, kteří jen provedou ohodnocení projektu podle metodiky založené na metodě UCP, která je v softwarovém průmyslu běžně používána. O výpočet odhadů se postará framework popsaný v této práci. Dále je možné implementovat tento framework pro odhady v určité softwarové společnosti, ale i pro velké datasety. Firmy tak nemusí přebírat zažité a obecné rovnice, ale může si sama vygenerovat odhadovací rovnici přesně podle svých historických projektů.

8 ZÁVĚR

Tato práce pojednává o novém frameworku odhadování časových náročností softwarových projektů. Získávání přesnějších odhadů je jednou z kritických částí cyklu softwarového vývoje. Tato práce jako celek má přispět k vytvoření frameworku pro přesnější odhad tohoto úsilí pomocí metod symbolické regrese.

Nový framework je založen na principu generování matematických modelů. Generování matematických modelů probíhá pomocí algoritmu analytického programování a diferenciální evoluce.

Pro odhad základních parametrů projektu je pro daný výzkum použita metoda UCP. Ta je vhodná zejména díky jednoduchosti odhadu těchto parametrů a její relativní rozšířenosti pro odhadování projektů obecně. Metoda UCP poskytuje tyto základní parametry projektu v raných fázích vývojového cyklu.

V práci jsou řešeny zejména vlastnosti a nastavení tohoto nového frameworku, tak aby poskytoval zpřesnění odhadů časového úsilí. Byly řešeny otázky linearity mezi odhadem a skutečným časovým úsilím, dále možnosti optimalizace frameworku, jak pomocí snížení velikosti prohledávaného prostoru, tak změnou účelové funkce.

Framework byl testován na dvou datasetech s využitím křížové validace. Datasety byly získány z vědeckých článků a jiné odborné literatury, kde prezentovaný algoritmus dosahuje zpřesnění oproti standardní Karnerově UCP rovnici obecně až o 18 %.

Na datasetu s menším počtem vzorků dosahuje prezentovaný framework průměrné relativní chyby 40 %, což je zpřesnění oproti ostatním použitým metodám v průměru až 20 % a oproti standardní UCP rovnici až o 18 %. Na větším datasetu se průměrná relativní chyba pohybuje na hodnotě 8 %, která je srovnatelná s ostatními použitými metodami a taktéž zpřesněním oproti UCP rovnici až o 18 %.

LITERATURA

[1] Albrecht, A. J. Measuring application development productivity, 1979.

[2] Beckert, Bernd; Schleife, Katrin; Dupuis, Dominique; Wydra, Sven; Niemann, F. The economic and social impact of software & services on competitiveness and innovation. Technical report, 2017.

[3] Bleuler, S., Brack, M., Thiele, L. and Zitzler, E. Multiobjective genetic programming: reducing bloat using SPEA2. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. 2001, 1, pp. 536–543.

[4] Boehm, W. Software Engineering Economics. *IEEE Transactions on Software Engineering*. jan 1984, SE-10, 1, pp. 4–21. ISSN 0098-5589.

[5] Briand, L. C., El Emam, K. and Bomarius, F. COBRA: a hybrid method for software cost estimation, benchmarking, and risk assessment. In *Proceedings of the 20th International Conference on Software Engineering*, 1998. ISBN 0-8186-8368-6.

[6] Burgess, C. J., Lefley, M. and Le, M. Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*. 2001, 43, 14, pp. 863–873. ISSN 09505849 (ISSN).

[7] Collector, D. and Module, F. G. Qualitative Research Methods Overview. *Qualitative Research Methods A Data Collectors Field Guide*. 2011. ISSN 00222437.

[8] Doshi-Velez, F. and Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. 2017, , Ml, pp. 1–13.

[9] Hughes, R. T. Expert judgement as an estimating method. *Information and Software Technology*. jan 1996, 38, 2, pp. 67–75. ISSN 09505849.

[10] Karner, G. Resource estimation for objectory projects. *Objective Systems SF AB*. 1993, pp. 1–9.

[11] Keung, J. W. Theoretical Maximum Prediction Accuracy for Analogy-Based Software Cost Estimation. *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*. 2008, pp. 495–502. ISSN 1530-1362.

[12] Kocaguneli, E. and Menzies, T. Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software*. 2013, 86, 7, pp. 1879–1890. ISSN 01641212.

[13] Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press., 1992.

[14] Krishnan, M. Against Interpretability: a Critical Examination of the Interpretability Problem in Machine Learning. *Philosophy and Technology*. 2019. ISSN 22105441.

[15] Lefley, M. and Shepperd, M. J. Using genetic programming to improve software effort estimation based on general data sets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2724, pp. 2477–2487, 2003.

[16] Ochodek, M., Nawrocki, J. and Kwarciak, K. Simplifying effort estimation based on Use Case Points. *Information and Software Technology*. mar 2011, 53, 3, pp. 200–213. ISSN 09505849.

[17] Oplatkova, Z., Viktorin, A., Senkerik, R. and Urbanek, T. Different approaches for constant estimation in analytic programming. In *Proceedings - 31st European Conference on Modelling and Simulation, ECMS 2017*, 2017. ISBN 9780993244049.

[18] Shan, Y., McKay, R., Lokan, C. and Essam, D. Software project effort estimation using genetic programming. *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*. 2002, 2, pp. 1108–1112.

[19] Sheta, A. F. and Al-Afeef, A. Software Effort Estimation for {NASA} Projects Using Genetic Programming. *Journal of Intelligent Computing*. 2010, 1, 3, pp. 146–156. ISSN 0976-9005.

[20] Silhavy, R., Silhavy, P. and Prokopova, Z. Automatic complexity estimation based on requirements. In *Latest Trends on Systems*, II, pp. 4, Santorini, Greece, 2014. ISBN 9781618042446.

[21] Silhavy, R., Silhavy, P. and Prokopova, Z. Algorithmic Optimisation Method for Improving Use Case Points Estimation. *PLOS ONE*. nov 2015, 10, 11. ISSN 1932-6203.

[22] Storn, R. and Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*. 1997, pp. 341–359. ISSN 0925-5001.

[23] Subriadi, A. P. and Ningrum, P. A. Critical review of the effort rate value in use case point method for estimating software development effort. *Journal of Theroretical and Applied Information Technology*. 2014, 59, 3, pp. 735–744.

[24] Trendowicz, A. and Jeffery, R. *Software Project Effort Estimation*. 2014. ISBN 9783319036281.

[25] Urbánek, T. *Objevování trigonometrických identit pomocí umělé inteligence*. diplomová práce (Ing.), Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, 2011.

[26] White, H. and Sabarwal, S. Quasi-experimental design and methods. *Methodological Briefs: Impact Evaluation 8, UNICEF Ofice of Research, Florence*. 2014.

[27] Zelinka, I., Oplatkova, Z. and Nolle, L. Analytic programming - Symbolic Regression by means of arbitrary Evolutionary Algorithms. *International Journal of Simulation Systems, Science & Technology*. 2005, 6, pp. 44–56. ISSN 1473-8031.

[28] Zelinka, I., Davendra, D., Senkerik, R., Jasek, R. and Oplatkova, Z. *Analytical programming-a novel approach for evolutionary synthesis of symbolic structures*. InTech, 2011. ISBN 978-953-307-171-8.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AP	Analytické programování
COCOMO	Constructive Cost Model
DE	Diferenciální evoluce
ECF	Environmental complexity factor
FP	Function point
GFS	General function set
LAD	Least absolut deviation
LASSO	Least absolute shrinkage and selection operator
MMRE	Mean Magnitude of Relative Error
MSE	Mean Squared Error
OLS	Ordinary least square
SOMA	Self-organizing migration algorithm
TCF	Technical complexity factor
UAW	Unadjusted actor weights
UML	Unified modeling language
UUCW	Unadjusted use case weights

PUBLIKAČNÍ AKTIVITY AUTORA

[1] T. Urbanek, Z. Prokopova, and R. Silhavy. On the usage of differential evolution for effort estimation. In *18th International Conference on Circuits (part of CSCC ´14)*, pages 632–635, Santorini Island, Greece, 2014.

[2] T. Urbanek, Z. Prokopova, R. Silhavy, and S. Sehnalek. Using Analytical Programming and UCP Method for Effort Estimation. In *Modern Trends and Techniques in Computer Science*, volume 285, pages 571–581. Springer International Publishing, 2014.

[3] T. Urbanek, Z. Prokopova, and R. Silhavy. *On the value of parameters of use case points method*, volume 347. 2015.

[4] T. Urbanek, Z. Prokopova, R. Silhavy, and V. Veselá. Prediction accuracy measurements as a fitness function for software effort estimation. *SpringerPlus*, 2015.

[5] T. Urbanek, Z. Prokopova, R. Silhavy, and A. Kuncar. New Approach of Constant Resolving of Analytical Programming. *30th European Conference on Modeling and Simulation*, pages 231–236, 2016.

[6] T. Urbanek, Z. Prokopova, R. Silhavy, and A. Kuncar. Using improved analytical programming algorithm for effort estimation in software engineering. In *MATEC Web of Conferences*, volume 76, 2016.

[7] T. Urbanek, Z. Prokopova, R. Silhavy, and A. Kuncar. Using Analytical Programming for Software Effort Estimation. volume 465, pages 261–272. 2016.

[8] T. Urbanek, Z. Prokopova, R. Silhavy, and A. Kuncar. Using improved analytical programming algorithm for effort estimation in software engineering. *MATEC Web of Conferences*, 76:02009, oct 2016.

[9] A. Kuncar, M. Sysel, and T. Urbanek. Calibration of low-cost triaxial magnetometer. In *MATEC Web of Conferences*, volume 76, 2016.

[10] A. Kuncar, M. Sysel, and T. Urbanek. Calibration of low-cost triaxial magnetometer. *MATEC Web of Conferences*, 76:05008, oct 2016.

[11] T. Urbanek, A. Kolcavova, and A. Kuncar. Inferring productivity factor for use case point method. In *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, 2017.

[12] A. Kuncar, M. Sysel, and T. Urbanek. *Calibration of low-cost three axis accelerometer with differential evolution*, volume 573. 2017.

[13] A. Kuncar, M. Sysel, and T. Urbanek. Calibration of low-cost accelerometer and magnetometer with differential evolution. In *ICMT 2017 - 6th International Conference on Military Technologies*, 2017.

[14] Z.K. Oplatkova, A. Viktorin, R. Senkerik, and T. Urbanek. Different approaches for constant estimation in analytic programming. In *Proceedings - 31st European Conference on Modelling and Simulation, ECMS 2017*, 2017.

[15] A. Kuncar, M. Sysel, and T. Urbanek. Differential evolution as calibration technique for three axis gyroscope. In *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, 2017.

[16] J. Vychytilová, D. Pavelková, H. Pham, and T. Urbánek. Macroeconomic factors explaining stock volatility: multi-country empirical evidence from the auto industry. *Economic Research-Ekonomska Istrazivanja*, 32(1):3327–3341, 2019.

[17] J. Švarcová, T. Urbánek, L. Povolná, and E. Sobotková. Implementation of r & d results and industry 4.0 influenced by selected macroeconomic indicators. *Applied Sciences (Switzerland)*, 9(9), 2019.

ODBORNÝ ŽIVOTOPIS AUTORA

OSOBNÍ ÚDAJE

Urbánek Tomáš Ing.
Kvítková 690, 760 01 Zlín (Česká republika)
Tel: +420 728 695 707
E-mail: t.urbanek@email.cz
Pohlaví: Muž
Datum narození: 1.4.1987
Státní příslušnost: Česká

PRACOVNÍ ZKUŠENOSTI

09/2011–05/2012 Computer programmer
Cominfo, a.s., Zlín (Česká republika)

01/09/2017–do současnosti Akademický pracovník UTB, Zlín
Tajemník ústavu Statistiky a kvantitativních metod

VZDĚLÁNÍ, ODBORNÁ PŘÍPRAVA A KURZY

27/11/2013–24/12/2013 Zahraniční stáž : Francie, Polytech Lille
2019-2020 Zahraniční výuka : Irácký Kurdistán

2013–do současnosti Doktorské studium
Univerzita Tomáše Bati ve Zlíně - Fakulta aplikované informatiky, Zlín
Inženýrská informatika
2009–2011 Magisterské studium
Univerzita Tomáše Bati ve Zlíně - Fakulta aplikované informatiky, Zlín
Inženýrská informatika
2006–2009 Bakalářské studium
Univerzita Tomáše Bati ve Zlíně - Fakulta aplikované informatiky, Zlín
Inženýrská informatika
2002–2006 Maturita
Střední průmyslová škola a Obchodní akademie Uherský Brod, Uherský Brod
Mechanik - Elektronik zaměření na výpočetní techniku

OSOBNÍ DOVEDNOSTI

Mateřský jazyk : čeština
Další jazyky : angličtina B2

Certifikát GOPAS : Základy UML jazyka
Kancelářské balíky : Microsoft Office, Libre Office
Databázové systémy : MySQL, Oracle, CauchDB
Programovací jazyky : R, C/C++, C#, Python, PHP, Javascript, Lua, Lisp
Vědecký software : Mathematica, R, Matlab
Grafické aplikace : Gimp, Inkscape
Publikační software : Latex, Mendeley
Specializovaný statistický software : R, Stan, JAGS, WebPPL, Church

DOPLŇUJÍCÍ INFORMACE

Projekty Hlavní řešitel :
IGA/FAI/2013/032 Využití evolučních algoritmů ke zpřesnění odhadů časových náročností softwarových produktů
IGA/FAI/2014/019 Optimalizace evolučních algoritmů při jejich využití k odhadování časových náročností softwarových projektů
IGA/CebiaTech/2015/034 Vývoj webové aplikace podporující princip RAD pro sběr dat a výzkum metody odhadování časové náročnosti softwarových projektů pomocí analytického programování

Spoluřešitel:
IGA/FAI/2016/035 Kalibrace MEMS senzorů
IGA/FAI/2017/007 Sensor fusion
TAČR/ZETA/TJ01000142

Tomáš Urbánek

Využití analytického programování pro odhady časových náročností vývoje softwarových projektů

Usage of analytical programming for effort estimation of software projects

Teze dizertační práce

Vydala Univerzita Tomáše Bati ve Zlíně

nám. T. G. Masaryka 5555, 760 01 Zlín

První vydání

Náklad: vyšlo elektronicky

Sazba: Ing. Tomáš Urbánek, Ph.D.

Publikace neprošla jazykovou ani redakční úpravou.

Rok vydání: 2020

ISBN 978-80-7454-975-5