Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Disertační 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**
Obor:	Inženýrská informatika
Školitel:	doc. Ing. Zdenka Prokopová, CSc.
Konzultant:	doc. Ing. Radek Šilhavý, Ph.D.

Zlín, 2020

„Vědět je uznat, když něco víš, že to víš, a když něco nevíš, uznat, že to nevíš.“
Konfucius

Rád bych poděkoval vedoucí této práce paní doc. Ing. Zdence Prokopové, CSc. a konzultantovi doc. Ing. Radku Šilhavému, Ph.D. za čas, který mi ochotně věnovaly při konzultacích a za předávání cenných rad, které mi vždy rádi poskytly.

Chtěl bych také poděkovat svým rodičům, kteří mi poskytli nutné zázemí a bez nichž by tato práce nemohla nikdy vzniknout. Děkuji také své přítelkyni Adélce, která mi poskytla podporu a prostor, který jsem pro tuto práci potřeboval. Snad budu moci všem toto strádání v budoucnu nahradit.

ABSTRAKT

Disertační práce je zaměřena na výzkum v oblasti softwarového inženýrství specificky na získávání odhadů časového úsilí. 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. 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. 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 %.

SUMMARY

This thesis is focused on research in the field of software engineering; specifically on obtaining estimates of time effort. Obtaining more accurate estimates is one of the critical parts of the software development cycle. 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. This work deals mainly with the properties and settings of this new framework, so as to provide a refinement of estimates of time effort. 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

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

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. [6]. 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 [48]. A je jednou ze základních částí tak zvané podpory softwarového procesu [22, 50, 113].

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 [109]. 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.

1.1 Motivace a hlavní myšlenky disertační práce

Z článků The Chaos Report [105] vyplývá, že přibližně 43 % vyvíjeného softwaru je vyvinuto se zpožděním a nebo s překročením rozpočtu, proto je nesmírně důležité se této části softwarového inženýrství věnovat. Souvislost s těmito alarmujícími čísly může mít právě i odhad úsilí. Odhad úsilí je jedním z nástrojů pro správné rozhodování v době vývoje softwaru. Softwarový manažer může na základě nesprávného odhadu činit nesprávná rozhodnutí, což může vést minimálně k překročení rozpočtu.

Hlavní motivací pro tento výzkum je tedy vytvořit framework, který pomůže projektovým manažerům odhadovat toto časové úsilí přesněji a nebude klást zvýšené nároky na jejich znalosti v oblasti statistiky a strojového učení.

Další myšlenkou této práce je důležitost odhadu úsilí v raných fázích vývoje softwaru, nejlépe v době analýzy požadavků [47]. Z tohoto důvodu byl pro tento výzkum použit způsob výpočtu parametrů projektu založený na metodě Use Case Points (dále UCP). Toto řešení poskytne odhad parametrů projektu v raných fázích vývojového cyklu. Tyto odhady parametrů projektu budou dále zpracovány frameworkem prezentovaným v této práci.

Vývoj softwaru je velmi komplexní činností, která zahrnuje spoustu faktorů jako je např. velikost vývojového týmu, použitý programovací jazyk, zkušenosti vývojového týmu s danou problematikou atd. V reálném vývoji nemusejí být vztahy mezi odhadem a dokončením softwarového systému vždy lineární a zahrnují velké množství neurčitostí. Na základě této myšlenky musí být v navrhovaném frameworku zahrnuty metody, které dokáží vytvořit matematické modely s nelineárními prvky a zároveň se model musí vypořádat s jistým množstvím neurčitosti. Proto použití vhodných metod umělé inteligence, jako je např. analytické programování, může být cestou ke zpřesňování odhadů časových náročností. Práce si klade za cíl vytvořit nový framework, který pomocí metod umělé inteligence umožní pro uživatele zjednodušit a hlavně zpřesnit odhady časových náročností softwarových projektů.

2 Cíle disertační práce

V následující kapitole budou definovány hlavní a dílčí podpůrné cíle disertační práce vycházející z její motivace a hlavních myšlenek. Realizace hlavního cíle disertační práce předpokládá naplnění dílčích podpůrných cílů disertační práce.

2.1 Hlavní cíl

- Návrh frameworku k získání přesnějších odhadů časových náročností softwarových projektů

2.2 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

V návaznosti na teoretická východiska získaná kritickou literární rešerší dostupných zdrojů byly vytvořeny jednotlivé výzkumné otázky a postupy k jejich naplnění.

2.3 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

Na základě těchto výzkumných otázek budou dále sestaveny a otestovány vědecké hypotézy. Hypotézy jsou definovány v kapitole 6 zvlášť u každé výzkumné otázky.

3 Teoretická rešerše

Jenkins et al. [45] zpracovali velkou empirickou studii zaměřenou na odhad časového úsilí v raných fázích vývojového cyklu. Studie Heemstra et al. [38] dala základy toho jak a kdy je nutné ohodnocovat softwarové projekty. Obě studie dokládají, že je nutné ohodnocovat softwarové projekty v raných fázích vývoje softwaru. Studie Jorgensen et al. [46] se zaměřuje na vylepšování odhadů časových náročností prováděním systematických revizí. Studie Satapathy et al. [89] a Xu et al. [119] ukazují, že metoda Function Point podává nekonzistentní odhady. Studie Nassif et al. [62] využívá k ranému odhadu kaskádově korelované neuronové sítě s využitím Use Case Points metody. Studie Demirors et al. [23] ukazuje, že průměrná odchylka mezi odhadovanou a skutečnou časovou náročností za použití function point analýzy je přibližně 10 %. Současně studie Pengelly [75] ukazuje, že obdobnou odchylku má i metoda COCOMO, avšak metoda je závislá na mnoha proměnných.

Studie [13, 17, 93, 58, 91, 110] testovaly efektivitu používání symbolické regrese pro zpřesňování časového úsilí. Tyto studie mají z funkčního hlediska blízko právě k metodě analytického programování, o které pojednává tato disertační práce.

Výzkum Burgess et al. [13] používá pro odhady velmi podobnou techniku jako v této práci. Autoři používají pro symbolickou regresi algoritmus Genetického programování [54, 55] a pro odhad základních parametrů projektu je použita metoda COCOMO. Autoři studie se s efektem "bloat" [8] vznikajícím právě v algoritmu Genetického programování vypořádali tak, že řešení s hlubokými stromy zahazovali. Mohlo tedy docházet k odstraňování potenciálně lepších řešení u stromů s větší hloubkou. Jejich závěr je, že tyto metody jsou použitelné, ale jejich nastavení je příliš komplexní. Závěry ohledně použitelnosti těchto metod potvrdila také studie Sheta et al. [93], 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. [58]. 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. [91] 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 [56].

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. Interpretovatelnost modelu je využita zejména pro analýzu řešení, kdy se dozvíme jakým způsobem se dospělo k výsledku. Interpretovatelný model může být snadno zkontrolován, auditován, a tím roste důvěra v použitý model [25]. Většina citovaných studií se potýká s problémy nastavení algoritmu genetického programování. Z tohoto důvodu byl pro tuto práci použit algoritmus analytického programování [122], 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 [8]. Analytické programování tímto efektem netrpí. V celém výzkumu je pro posouzení kvality řešení použito zejména kritérium MMRE. Hlavním důvodem je jeho rozšíření a ve vědeckých publikacích v tomto oboru je téměř vždy uváděn spolu s dalšími.

V současné době je výzkum problematiky zpřesňování odhadů časových náročností softwarových projektů velmi aktuální, což je způsobeno hlavně poptávkou po softwarových řešeních a produktech. Přínos přesnějších odhadů pak spočívá ve vytváření efektivnějších plánů pro řízení softwarového vývoje. Hlavní motivací je odstranění lidského faktoru z tohoto procesu, a tím vytvoření přesnějších výpočtů časových náročností. Proto techniky využívající umělou inteligenci hrají významnou roli. Většina prací se snaží vylepšit již zavedené a běžně používané metody jako je např. metoda Use Case Points, Function Point (dále FP) nebo Constructive Cost Model (dále COCOMO). Běžně se vyskytují stu-

die využívající neuronové sítě na odhad časových náročností. Štolfa et al. [100] využívají neuronové sítě s využitím fuzzy logiky. Vyčerpávající studie Dejaeger et al. [22] provedla srovnání podporujících technik pro odhad časových náročností s využitím metod umělé inteligence, kdy dospěly k názoru, že je velmi těžké komparativní porovnání za použití rozdílných technik hlavně pre-procesingu dat. Studie Cerpa et al. [14] pojednává o použití metod umělé inteligence pro klasifikaci v odhadování časových náročností.

4 Teoretická východiska disertační práce

Existuje mnoho metod pro odhadování časových náročností softwarových projektů. Z taxonomického hlediska můžeme metody pro odhadování rozdělit do tří hlavních kategorií [109, 63]:

- Metody řízené daty
- Expertní úsudek
- Hybridní metody

Někteří autoři zde zahrnují přímo i samostatné metody softcomputingu např. Štrba et al. [103]. 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 [9], FP [2, 5] a metoda UCP [47]. Podle Trendowicz et al. [109] je velkou výhodou těchto metod, že nevyžadují příliš velké zásahy experta do odhadů a jsou flexibilní. Nevýhodou však zůstává, že když nemáme k dispozici historické projekty není možné zkonstruovat odhad.

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 [42]. Často se také využívá analogií současného projektu s projektem řešeným v minulosti. Nejznámější metody z této kategorie jsou Delphi, Wideband Delphi [9], Planning Poker [34] 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) [12] či metody založené na Bayesově teorému.

Poslední skupinou jsou metody, které využívají přímo metod softcomputingu. Autor této disertační práce se však domnívá, že je možné tyto metody zařadit do jedné ze zmíněných kategorií a to nejčastěji metody řízené daty. Proto, popsanou metodu v této práci řadíme do metod řízených daty.

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 [52].

V následujících podkapitolách budou uvedeny metody použité pro tuto disertační práci.

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 [47] a jedná se o metodu, která se řadí z hlediska taxonomického do metod řízených daty [109]. Projektový manažér musí odhadnout 4 parametry projektu. Tyto parametry jsou následující:

- Neupravená váha případu užití (UUCW)
- Neupravená váha aktérů (UAW)
- Technický faktor (TCF)
- Faktor prostředí (ECF)

4.1.1 Neupravená váha případu užití

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. Všechny kategorie s váhami jsou prezentovány v tabulce 4.1. Vliv jednotlivých kategorii se řeší sumací a váhováním, jak vyplývá z rovnice 4.1.

Tab. 4.1 Neupravená váha případu užití

Klasifikace	Počet transakcí	Váha
Jednoduchý	1 až 3 transakce	5
Průměrný	4 až 7 transakcí	10
Složitý	8 a více transakcí	15

$$UUCW = \sum_{i=1}^{n} (c_i \times w_i), \tag{4.1}$$

kde n je počet případu užití, c_i je klasifikace í-tého případu užití a w_i je váha í-tého případu užití.

4.1.2 Neupravená váha aktérů

Je to parametr softwarového projektu zabývající se komplexitou aktérů vyvíjeného systému. Metoda UCP používá tři kategorie pro klasifikaci neupravené váhy aktérů. Všechny kategorie s váhami jsou prezentovány v tabulce 4.2. Vliv jednotlivých kategorií se řeší sumací a váhováním, jak vyplývá z rovnice 4.2.

Tab. 4.2 Neupravená váha aktérů

Aktér Klasifikace	Typ	Váha
Jednoduchý	Externí systém s definovaným API	1
Průměrný	Externí systém komunikující standardním protokolem (příklad TCP/IP, FTP, ...)	2
Složitý	Člověk používající GUI rozhraní	3

$$UAW = \sum_{i=1}^{n} (c_i \times w_i), \tag{4.2}$$

kde n je počet aktéru, c_i je klasifikace í-tého aktéra a w_i je váha í-tého aktéra.

4.1.3 Technický faktor

Metoda UCP ohodnocuje softwarový projekt třinácti technickými faktory. Tyto se zaměřují zejména na technickou komplexitu vyvíjeného systému. Všechny technické faktory jsou prezentovány v tabulce 4.3. 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. Výpočet je prezentován rovnicí 4.3.

Tab. 4.3 Technické faktory

Faktor	Popis	Hodnota	Váha
T1	Distribuované zpracování	$< 0; 5 >$	2,0
T2	Výkonnost	$< 0; 5 >$	1,0
T3	Efektivita uživatele	$< 0; 5 >$	1,0
T4	Složitost operací	$< 0; 5 >$	1,0
T5	Znovupoužitelnost kódu	$< 0; 5 >$	1,0
T6	Jednoduchost instalace	$< 0; 5 >$	0,5
T7	Jednoduchost užití	$< 0; 5 >$	0,5
T8	Přenositelnost	$< 0; 5 >$	2,0
T9	Usnadnění změny	$< 0; 5 >$	1,0
T10	Souběžnost	$< 0; 5 >$	1,0
T11	Speciální požadavky na bezpečnost	$< 0; 5 >$	1,0
T12	Přímé zapojení třetí strany	$< 0; 5 >$	1,0
T13	Trénink uživatelů	$< 0; 5 >$	1,0

$$TCF = 0,6 + (0,01 \times \sum_{i=1}^{13} (v_i \times w_i)), \tag{4.3}$$

kde v_i je hodnota í-tého technického faktoru a w_i je váha í-tého technického faktoru.

4.1.4 Faktor prostředí

Metoda UCP obsahuje 8 faktorů prostředí. Tyto se zaměřují na komplexitu prostředí vyvíjeného systému. Všechny faktory prostředí jsou prezentovány v tabulce 4.4. 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. Výpočet je vidět na rovnici 4.4.

Tab. 4.4 Faktory prostředí

Faktor	Popis	Hodnota	Váha
E1	Obeznámení s projektem	$< 0; 5 >$	1,5
E2	Zkušenosti s aplikacemi	$< 0; 5 >$	0,5
E3	Zkušenosti s objektově orientovaným programováním	$< 0; 5 >$	1,0
E4	Kvalifikace vedoucího analytika	$< 0; 5 >$	0,5
E5	Motivace	$< 0; 5 >$	1,0
E6	Stálost požadavků	$< 0; 5 >$	2,0
E7	Zaměstnanci na částečný úvazek	$< 0; 5 >$	-1,0
E8	Složitost programovacího jazyka	$< 0; 5 >$	-1,0

$$ECF = 1,4 + (-0,03 \times \sum_{i=1}^{8} (v_i \times w_i)), \tag{4.4}$$

kde v_i je hodnota í-tého faktoru prostředí a w_i jeho váhy.

4.1.5 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 [47] a jeho hodnota byla vypočtena na 20 hodin na jeden bod případu užití. Nicméně stanovení této hodnoty se věnuje řada studií. Studie Schneider a Winters [90] doporučují používat tří hodnoty na základě vyhodnocení environmentálního faktoru 20, 28 a 36 člověkohodin na jeden bod

UCP. Studie Nassif et al. [65] navrhla nelineární výpočet faktoru produktivity s využitím fuzzy modelu. Obecně se však tato hodnota pohybuje v intervalu $< 15; 30 >$ [104].

4.1.6 Výpočet odhadu

Pro kalkulaci počtu bodů případu užití je používána rovnice 4.5.

$$UCP = (UUCW + UAW) \times TCF \times ECF, \tag{4.5}$$

kde UCP je počet bodů případu užití.

Pro kalkulaci výsledného odhadu je používána rovnice 4.6.

$$EE = UCP \times FP, \tag{4.6}$$

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

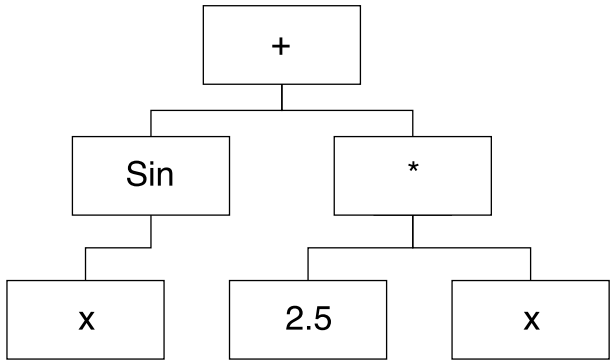
4.2 Symbolická regrese

V této práci je použita metoda tzv. symbolické regrese pro zpřesnění odhadů časových náročností softwarových projektů. 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 symboly
- Terminální symboly

Neterminální symboly jsou funkce, ať už matematické, či uživatelsky definované (realizované formou zdrojového kódu). Každá funkce, ať už matematická



Obr. 4.1 Příklad vytvoření funkce $f(x) = sin(x) + 2,5x$

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.

Obrázek 4.1 zobrazuje datovou strukturu stromu pro funkci $f(x) = sin(x) + 2,5x$. Kde funkce $Sin(\cdot)$, $*(\cdot,\cdot)$ a $+(\cdot,\cdot)$ představují neterminální symboly. Konstanta 2,5 a proměnná x představují terminální symboly. Takto reprezentované funkce se vyskytují v algoritmu genetického programovaní, ale obecně lze takto chápat jakoukoliv funkci.

Metody používané pro symbolickou regresi jsou následující:

- Genetické programování
- Gramatická evoluce
- Analytické programování

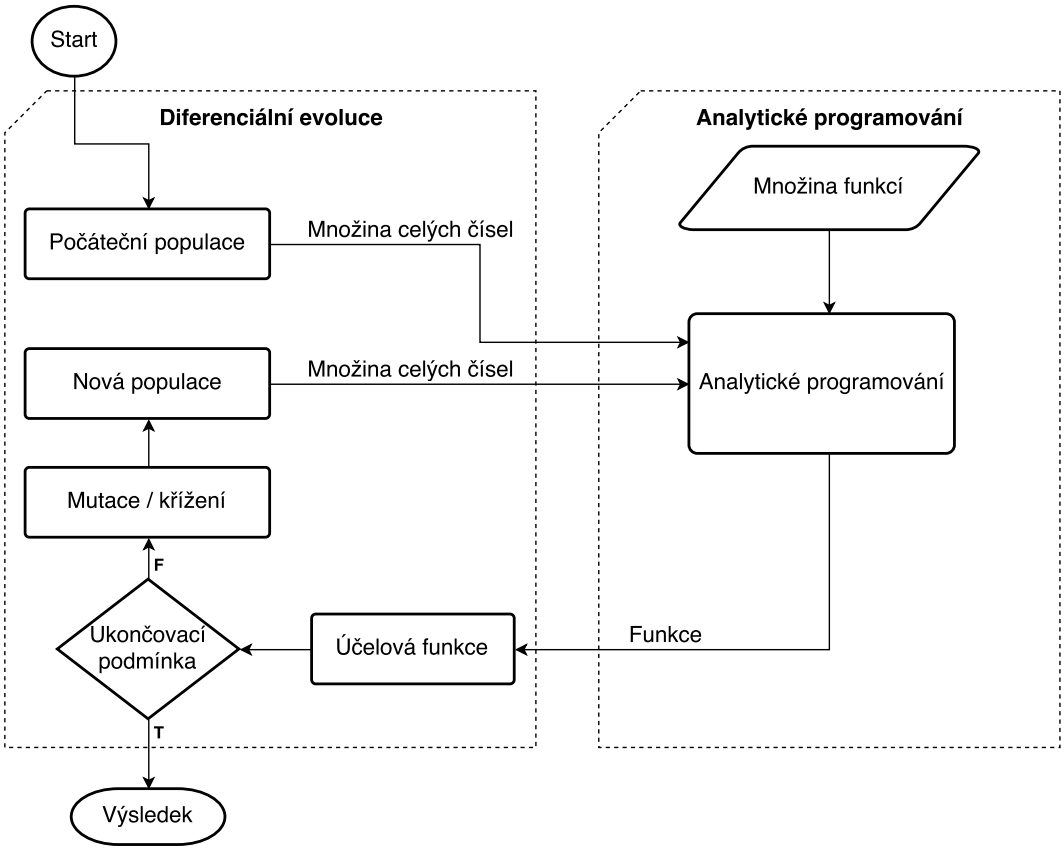
Metoda použitá pro tuto práci se nazývá Analytické programování [122]. Genetické programovaní [54, 55] jako možná metoda pro tuto práci nevyhovuje hlavně z důvodu efektu tzv. bobtnání programu (bloat). Efekt "bloat" vzniká když začne být řešení optimalizačního problémů příliš komplexní resp. dochází

k tzv. overfittingu. Jestliže se na řešení optimalizačního problému budeme dívat z perspektivy datové struktury stromu, který je řešením daného problému, pak se efekt bloat projevuje jako velmi hluboký strom. U takových hlubokých stromů se pak stává, že je výpočet zastaven (např. dělení nulou) nebo je součástí řešení např. násobení nulou (plýtvání výpočetního výkonu). Existuje samozřejmě mnoho technik a algoritmů, které tomuto efektu mohou zabránit, např. [8]. Nicméně Analytické programování tímto efektem netrpí už v samotném jeho návrhu. Ošetření efektu bloat se zde ošetřuje počtem použitých symbolů. Gramatická evoluce [86] využívá pro svou práci gramatiku reprezentovanou ve většině případů Backus-Naurovou formou [51]. Nevýhoda gramatické evoluce pro tuto práci spočívá v tom, že je potřeba sestavit gramatiku pro určitý programovací jazyk. Obecně se gramatická evoluce hodí spíše pro syntézu komplexních uživatelských programů a expertních systémů. V této práci se věnujeme pouze syntéze matematických popisů. Analytické programování bylo použito také z důvodů, že není závislé na použitém programovacím jazyku ani na použitém evolučním algoritmu, čímž vzniká modularita návrhu.

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 ter-

minálních a neterminálních symbolů. Do účelové funkce tak vstupuje sestavený jedinec dle celočíselné reprezentace z tabulky symbolů. Jakmile je jedinec sestaven, je ohodnocen účelovou funkcí některého z použitých evolučních algoritmů, např. SOMA nebo DE. Jak již bylo naznačeno výše, analytické programování je určitým druhem transformace z celočíselného jedince na sestavený výraz pomocí terminálních a neterminálních symbolů [123, 71, 112].



Obr. 4.2 Schéma práce analytického programování s diferenciální evolucí

Princip funkce analytického programování je prezentován na obrázku 4.2. Pro ilustrační účely používáme evoluční algoritmus diferenciální evoluce. Nejprve je generována počáteční populace. Tato populace sestává z přirozených čísel, která budou sloužit jako ukazatele do tabulky GFS. Analytické programování poté vytvoří funkci na základě této populace. Následně je tato vytvořená funkce ohod-

nocena tzv. účelovou funkcí. Jestliže je splněna ukončovací podmínka evolučního algoritmu, algoritmus končí. Pokud není ukončovací podmínka splněna, diferenciální evoluce vytvoří novou populaci pomocí mutace a křížení předešlých populací. Poté se celý proces opakuje s novou populací, až je dosaženo ukončovací podmínky. Po skončení algoritmu předpokládáme, že výsledné funkce, by měla být optimální, ale v mnoha případech suboptimálním řešením pro danou úlohu.

4.2.2 Diferenciální evoluce

Diferenciální evoluce vznikla v roce 1995 a jejími autory jsou Ken Price a Rainer Storm [102, 101]. 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 - parametr udává velikost vytvořené populace $NP \geq 4$
- F - parametr mutační konstanty $F \in < 0; 2 >$
- CR - určuje křížení v populaci a nazývá se práh křížení $CR \in < 0; 1 >$
- Generations - parametr určuje, kolik proběhne iterací algoritmu než bude ukončen

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 4.7.

$$v_j = x^G_{r3,j} + F \cdot \left(x^G_{r_1,j} - x^G_{r2,j}\right), \tag{4.7}$$

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.

U diferenciální evoluce je nutné dávat si pozor na jev zvaný „stagnace“. Stagnace je jednou z největších nevýhod diferenciální evoluce. V podstatě jde o zastavení vývoje hodnot účelové funkce k nižším hodnotám, než je dosaženo globálního extrému. Aby bylo možné stagnaci minimalizovat, je třeba věnovat pozornost nastavení parametrů diferenciální evoluce [112].

4.3 Metody použité pro komparaci

V této kapitole bude uveden krátký popis metod použitých pro komparaci s metodou prezentovanou v této práci.

4.3.1 Obecný lineární model a mnohonásobná lineární regrese

Lineární regrese je matematická metoda, která se používá k proložení bodů v grafu přímkou. Řeší se vztah mezi závislou proměnnou a nezávislými proměnnými. Přičemž se předpokládá, že tento vztah bude lineární [31]. Úkolem lineární

regrese je najít přímku, která bude minimalizovat součet druhých mocnin odchylek neboli reziduí. Lineární regresi lze zobecnit na obecný lineární model a prokládat jinou funkci než přímku nebo dokonce používat pro jiné než spojité závislé proměnné. Rovnici obecného lineárního modelu lze zapsat ve tvaru 4.8.

$$\mathbf{Y} = \mathbf{XB} + \mathbf{U}, \tag{4.8}$$

kde $\mathbf{Y}$ je matice závislých proměnných, $\mathbf{X}$ je matice nezávislých proměnných, $\mathbf{B}$ je matice koeficientů, které budou odhadovány a $\mathbf{U}$ je matice chyb.

V případě mnohonásobné lineární regrese je velikost vektoru Y_i rovna jedné, tedy máme pouze jednu závislou spojitou proměnnou a m nezávislých proměnných. Důležitým předpokladem použití lineární regrese jsou normálně rozdělená rezidua. Tato podmínka je jednou z pěti Gauss-Markovových podmínek [106].

Gauss-Markovovy podmínky:

1. Linearita v parametrech
2. Data musí být náhodně vybrána
3. Proměnné nesmí být "perfektně" korelovány mezi sebou
4. Proměnné nesmí být korelovány s náhodnou chybou
5. Rezidua musí vykazovat konstantní rozptyl

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [29, 65, 78, 57, 120].

4.3.2 Kroková lineární regrese

Pokud má regresní model větší počet nezávislých proměnných, může docházet k tomu, že existuje regresní model, který je jednodušší než model s větším počtem nezávislých proměnných. Kroková lineární regrese řeší právě tento případ. Jedná se v podstatě o postupný výběr a eliminaci nezávislých proměnných [39]. Postupy, které se u krokové regrese uplatňují jsou následující:

- Sestupný výběr
- Vzestupný výběr
- Výběr/eliminace

Sestupný výběr

Nejprve je spočten model se všemi nezávislými proměnnými. Následně se v každém kroku eliminuje nezávislá proměnná, která nejméně přispívá k vysvětlení závislé proměnné. V každém kroku dochází k výpočtům hypotéz, zda se koeficient rovná nule. Vyřazování proměnných končí až jsou všechny koeficienty u všech nezávislých proměnných statisticky významné.

Vzestupný výběr

Nejprve je spočten model, kde není obsažena žádná nezávislá proměnná. V každém kroku se k modelu přidává další nezávislá proměnná a testuje se, zda daný koeficient je statisticky významný. Jedná se v podstatě o opačný postup k sestupnému výběru. Zařazování nezávislých proměnných do modelu končí jakmile všechny zbylé nezařazené koeficienty nepřispívají k vysvětlení závislé proměnné.

Výběr/eliminace

Výběr/eliminace je v podstatě sloučením obou výše zmíněných metod do jednoho algoritmu. V jednom kroku dochází k zařazování a eliminaci nezávislých proměnných. Důvodem pro vznik této techniky je ten, že některé nezávislé proměnné vysvětlují závislou proměnnou pouze v kombinaci s jinou nezávislou proměnnou. Tato technika nám umožňuje provést právě tyto porovnání, ke kterým by u výše zmíněných metod nedošlo.

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [82, 94, 108, 66].

4.3.3 Ridge regrese

Ridge regrese někde také česky označována jako hřebenová regrese je metoda, která se používá, když mnohonásobná lineární regrese trpí problémem multikolinearity. Když nastává problém multikolinearity, tak metoda nejmenších čtverců nadále může poskytovat odhad s nejmenší systematickou chybou, ale rozptyly mohou dosahovat velkých hodnot a mohou být vzdálené od skutečné hodnoty. Ridge regrese řeší tento problém použitím metody nejmenších čtverců a přidáním penalizační konstanty λ. Penalizace se vypočítá použitím druhé mocniny hodnot sklonů. Tímto dochází k tomu, že odhad nadále nemá nejmenší systematickou chybu, ale může snížit rozptyl reziduí a tím zpřesnit odhad u modelů, které trpí problémem multikolinearity [76, 40].

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [73, 66].

4.3.4 LASSO regrese

Ve statistice je LASSO regrese jednou z regresních analýz, která provádí výběr proměnných a regularizaci koeficientů tak, aby bylo dosaženo přesnějších odhadů. Ridge regrese a LASSO regrese, jsou velmi podobné modely. LASSO regrese taktéž používá metody nejmenších čtverců a přidává penalizační konstantu λ. Penalizace se vypočítá použitím absolutních hodnot sklonů. Tímto dochází k tomu, že odhad nadále nemá nejmenší systematickou chybu, ale může snížit rozptyl reziduí a tím zpřesnit odhad. Důležitým rozdílem oproti Ridge regresi je možnost eliminace nezávislé proměnné, když je koeficient nulový a λ je relativně velké číslo [107, 36].

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [108, 66].

4.3.5 Rozhodovací strom CART

Rozhodovací strom je v oboru strojového učení model, který se používá pro regresi a klasifikaci. Pokud je rozhodovací strom určený pro regresi, tak se nazývá regresním stromem. Reprezentace rozhodovacího stromu je datová struktura bi-

nárního stromu. Každý uzel stromu reprezentuje jednu z nezávislých proměnných a hodnotu rozdělení. Každý list pak reprezentuje hodnotu závislé proměnné a představuje predikci. Naučený binární strom v podstatě představuje rozdělení prohledávaného 2D prostoru na obdélníkové části. Pro prostory o větší dimenzi než 3 se jedná o hyperkvádry [118, 80].

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [1, 120, 79, 61, 88].

4.3.6 K-nearest neighbors

Metoda k-nearest neighbors se používá zejména v oboru rozpoznávaní vzorů. Jedná se o neparametrickou metodu používanou pro klasifikační úkoly a regresi. Metoda funguje na principu nalezení k nejbližších bodech prohledávané plochy. V podstatě se jedna o lokální aproximaci funkce [3]. Jelikož je metoda k-nearest neighbors závislá na vzdálenosti jednotlivých bodů, tak normalizace předkládaných dat obvykle zvyšuje přesnost algoritmu [37]. Metoda k-nearest neighbors se liší v závislosti na úloze zadání.

Regrese

Vyhledá se k nejbližších bodů pro testovací hodnotu. Jakmile jsou tyto body nalezeny, tak se vypočítá průměrná hodnota těchto k nalezených bodů. Tento průměr je pak výsledkem regrese.

Klasifikace

Nejprve se vyhledá k nejbližších bodů ke klasifikovanému objektu. Poté se určí množství jednotlivých tříd do kterých těchto k objektů náleží. Klasifikovaný objekt je poté klasifikován do třídy, u které bylo zaznamenáno největší množství zařazení. Pokud je k nastaveno na hodnotu 1 pak je bod klasifikován do stejné třídy jako má nejbližší objekt k objektu klasifikovanému.

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [88, 69, 43].

4.3.7 Support vector machine

Support vector machine se používá zejména v oboru strojového učení ke klasifikačním a regresním úlohám. Algoritmus funguje na principu nalezení jasné hranice mezi klasifikovanými body a zároveň se snaží aby tato hranice byla co největší. Data, pro která získáváme predikci jsou potom zařazena do skupiny dle toho, na jakou stranu hranice náleží. Algoritmus funguje jen pouze jako učení s učitelem, tedy trénovací dataset musí mít předem danou třídu [11, 44].

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [121, 21, 68, 59].

4.3.8 Umělá neuronová síť

Umělé neuronové sítě jsou založeny na množství propojených jednotek zvaných umělé neurony. Tyto umělé neurony mají imitovat neurony, které jsou k nalezení v biologickém mozku [18]. Každé propojení mezi umělými neurony je nazýváno synapsí. Po synapsích jsou přenášeny hodnoty, které mají imitovat vzruchy. Do každého neuronu tedy vede určitý počet vstupních synapsí, které přenáší hodnoty. Výstupem z umělého neuronu jsou hodnoty, které jsou neuronem přepočítány, většinou formou nelineární funkce. Synapse v mnoha aplikacích obsahují i hodnotu tzv. váhy, která určuje jak moc bude signál/vzruch zesílen nebo zeslaben. Umělé neurony jsou uspořádány ve vrstvách a každá z těchto vrstev může provádět rozdílné matematické transformace. Signály pak putují z první vrstvy až do poslední vrstvy, kde očekáváme výsledek [26].

Neuronové sítě je nutné trénovat neboli "naučit". Pro učení neuronové sítě předkládáme vstupy a ideální výstup. Jedná se tedy o učení s učitelem. Pro každý z trénovacích vektoru jsou pak trénovacím algoritmem přenastaveny váhy synapsí. Jak moc budou přenastaveny váhy je funkcí rozdílu mezi výstupní hodnotou a ideální hodnotou. Jedním z nejznámějších trénovacích algoritmů pro umělé neuronové sítě je algoritmus backpropagation [41].

Pro odhadování časových náročností v softwarovém inženýrství byla tato metoda použita v publikacích [65, 81, 64, 72, 111, 97].

4.4 Problematika overfittingu

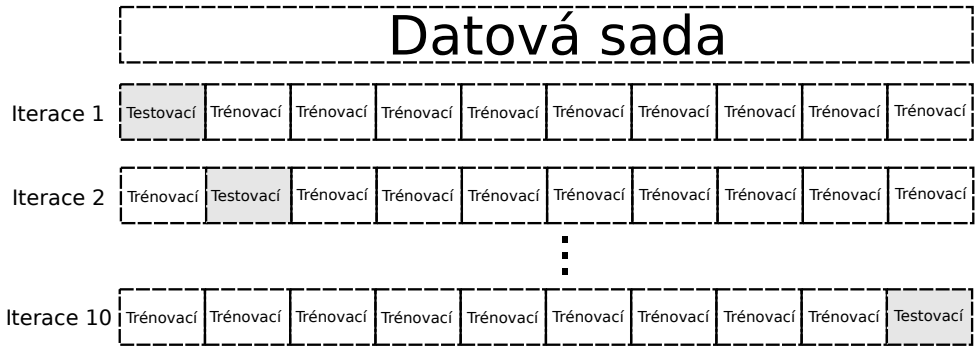
Pokud matematický model kopíruje data příliš těsně, je tento problém označován jako overfitting. Znamená to, že v matematickém modelu jsou odhadnuty parametry, které se příliš vážou na použitá data, a proto není model schopný odhadovat budoucí vstupy s dostatečnou přesností. Problém overfittingu nastává, když nemáme k dispozici nezávislá data k otestování finálního matematického modelu. Tento problém je většinou řešen rozdělením datové sady na trénovací množinu a na testovací množinu. Přístupy, které byly použity v tomto výzkumu jsou uvedeny v následujících kapitolách.

4.4.1 Náhodný výběr s opakováním

Tento přístup je založen na vybírání náhodných pozorování z použité datové sady. Mějme tedy datovou sadu o N pozorováních. Zvolíme, jak se má datová sada rozdělit na trénovací množinu a na testovací množinu. Doporučený počet pozorování, která vybíráme do testovací množiny, je obvykle kolem 33 % dostupných pozorování. Zbytek, tedy přibližně 66 %, pak bývá použit pro učení algoritmu. Pro příklad mějme 100 pozorování. Pokud tedy použijeme doporučené hodnoty, je 33 pozorování náhodně vyjmuto z datasetu a tato jsou použita pro testování naučeného algoritmu. Zbytek tedy 67 pozorování je použito pro naučení algoritmu. Tento postup je replikován x-krát, tím získáme x párů trénovacích a testovacích dat. Výhodou tohoto přístupu je zejména jednoduchá implementace, rychlost a také poměrně přesný odhad úspěšnosti, avšak v závislosti na dostatečném počtu opakování. Nevýhodou pak je, že se mohou data v trénovací sadě opakovat[37].

4.4.2 K-násobná křížová validace

U K-násobné křížové validace se datová sada rozdělí na k částí. Jedna z částí je vyjmuta a označena jako testovací množina, zbytek je pak použit pro učení algoritmu. Tento postup se opakuje tak dlouho, dokud nejsou použity všechny části pro testování. Příklad výběru dat pro 10-ti násobnou křížovou validaci je uveden na obrázku 4.3.



Obr. 4.3 Příklad výběru prvků 10-násobné křížové validace

Ve vědeckých studiích se nejčastěji volí hodnota $k = 10$. Obecně platí, že nižší hodnoty k vedou k přesnějšímu odhadu úspěšnosti, avšak mnohdy za cenu neúměrně vyšších výpočetních nároků [53]. Speciálním případem je $k = 1$, označován jako "leave-one-out". U tohoto případu je celá sada kromě jednoho pozorování použita na učení algoritmu. Toto jedno pozorování je pak použito pro otestování úspěšnosti. U velkých datových sad toto obvykle vede k velké výpočetní náročnosti. Právě tato velká výpočetní náročnost metody "leave-one-out" je hlavním důvodem k doporučované hodnotě $k = 10$. U větších hodnot k vzniká problém, že může záležet na tom, jak jsou data v jednotlivých částech seřazena [37]. Další argument pro volení vyšších hodnot k než 1 je poměr mezi varianci a systematickou chybou. Když je $k = 1$ tak má odhad velmi nízkou systematickou chybu a velkou varianci, což může vést k zavádějícím odhadům [87, 53].

4.5 Kritéria kvality

Mnoho vědeckých studií se zabývá také otázkou jaké zvolit kritérium pro kvalitní řešení. Tento problém se u symbolické regrese vyskytuje na dvou místech.

- Stanovení kritéria pro samotný běh algoritmu. Toto kritérium bývá označováno jako účelová funkce a určuje, o jak kvalitní řešení se jedná.
- Druhý výskyt měření kvality řešení se objevuje v konečné analýze, tedy v porovnávání, o jak moc kvalitní řešení se jedná v rámci již zpracovaného

výzkumu.

Vzhledem k velkému počtu různých kritérií kvality budou uvedeny pouze nejpoužívanější. Nejčastěji se objevující funkcí kvality finálního řešení v oboru softwarového inženýrství je funkce Mean Magnitude of Relative Error (dále MMRE). Avšak ke zvýšení vypovídající hodnoty výzkumu byly použity i další méně používané funkce jako je Least Absolute Deviation (LAD) nebo Mean Squared Error (MSE). V následujících kapitolách budou uvedeny jejich rovnice včetně popisu.

4.5.1 Kritérium kvality MMRE

Je nejpoužívanějším kritériem kvality modelu v oboru softwarového inženýrství. 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.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y_i}|}{y_i}, \tag{4.9}$$

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

Tato míra kvality je často kritizována [92, 49, 60] hlavně co se týče nesymetričnosti měření. Děje se tak v případech, kdy jsou jednotlivé položky $\frac{|y_i-\hat{y_i}|}{y_i}$ rovnice 4.9 výrazně šikmé. Pak je výsledek kritéria nespolehlivý. Problematické je také to, že se jedná o průměrnou hodnotu, kdy průměr je velmi náchylný na odlehlá pozorování. Autor Foss et al. [30] dokládá, že MMRE nevybere vždy nejlepší model, avšak podotýká, že zatím lepší kritérium kvality nemáme k dispozici. Autoři studie se domnívají, že by se mělo vycházet zejména z teoretického posouzení daného modelu. Autoři studie Port et al. [77] tvrdí, že ačkoliv je MMRE kritizováno je využíván téměř de facto jako standard pro porovnávání modelů v softwarovém inženýrství. Podobné tvrdí i studie Stensrud et al. [99]. Velkou výhodou MMRE je, bezpochyby, že se dají porovnat různé modely, jelikož výsledek je relativní chyba v procentech. I přes veškerou kritiku MMRE je nejvíce

využívaným kritériem v oblasti softwarového inženýrství a většina výzkumu v tomto oboru spoléhá právě na toto kritérium kvality.

4.5.2 Kritérium kvality LAD

Toto kritérium se využívá hlavně v oboru strojového učení pro nalezení optimálního řešení. Jedná se v podstatě o sumu odchylek od skutečné hodnoty. V této práci je toto kritérium využito hlavně jako účelová funkce k metodě diferenciální evoluce, avšak je použito také pro porovnání modelů. 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.

$$LAD = \sum_{i=1}^{n} |y_i - \hat{y}_i|, \tag{4.10}$$

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

Tato míra je také často porovnávána s podobnou mírou, a tou je OLS (ordinary least square). Jedinou změnou je výměna absolutní hodnoty za funkci druhé mocniny. LAD se používá také pro stejnojmennou statistickou metodu blízkou lineární regresi [98]. V této práci využíváme LAD častěji jako účelovou funkci z důvodu odolnosti vůči odlehlým hodnotám. Velkou nevýhodou pak je interpretace výsledné hodnoty, což je suma všech odchylek, v našem případě v člověkohodinách. Nehodí se také pro porovnávání různých metod, jelikož zachovává jednotku.

4.5.3 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. Mini-

malizací kritéria MSE dochází k minimalizaci rozptylu daného modelu. Funkce MSE je takzvaným druhým centrálním momentem chyby odchylek, který obsahuje jak rozptyly odchylek, tak jejich bias (systematická chyba ve sběru, analýze a interpretaci dat).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{4.11}$$

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

Nevýhodou použití MSE je, že funkce dává velkou váhu odlehlým pozorováním [7]. Toto se děje z důvodu použití druhé mocniny ve výpočtu a tedy odlehlé hodnoty jsou umocněny, což zvyšuje nepřesnost tohoto kritéria. Hlavní výhoda použití kriteria MSE spočívá v tom, že výsledkem je spojitá funkce konkrétně parabola. Oproti LAD je tedy možné nalézt minimum pomocí derivace. Této vlastnosti se využívá zejména v aplikacích lineární regrese.

5 Použité metody vědecké práce

Pro zpracování cílů disertační práce byly použity následující metody vědecké práce. Tyto metody byly použity podle jejich potřeby.

5.1 Metoda evaluace

Metoda evaluace je systematické posouzení kvality a hodnoty významu určitého objektu. Tato metoda je založena na důkladném sběru informací a na jejich odborném zpracování s cílem získat podklady pro případné rozhodnutí. V situaci, kdy je třeba řešit problém, poskytuje evaluace informace pro počáteční uvažování o možnostech řešení, a tak přispívá k formulaci konkrétního závěru [**?** 74]. Tato metoda bude použita pro studium vlastností jednotlivých výsledků algoritmu analytického programování pro vyjádření kvantitativního hodnocení.

5.2 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 [19, 20, 32]. 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.

5.3 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 [74, 116]. 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.

5.4 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.

5.5 Použitá metoda sběru dat

V disertační práci byla použita metoda sběru dat pomocí analýzy dostupných informačních zdrojů. Jedná se o využití sekundárních dat pro kvantitativní a kvalitativní výzkum získávaných z veřejně dostupných databází popř. prostudováním akademických studií. Popis dat a jejich získání je uveden v kapitole 6 strana 43.

6 Hlavní výsledky disertační práce

Následuje výčet hlavních výsledků, kterých autor docílil během svého doktorského studia. Jedná se především o návrh frameworku pro odhadování časových náročností softwarových projektů a simulační studii.

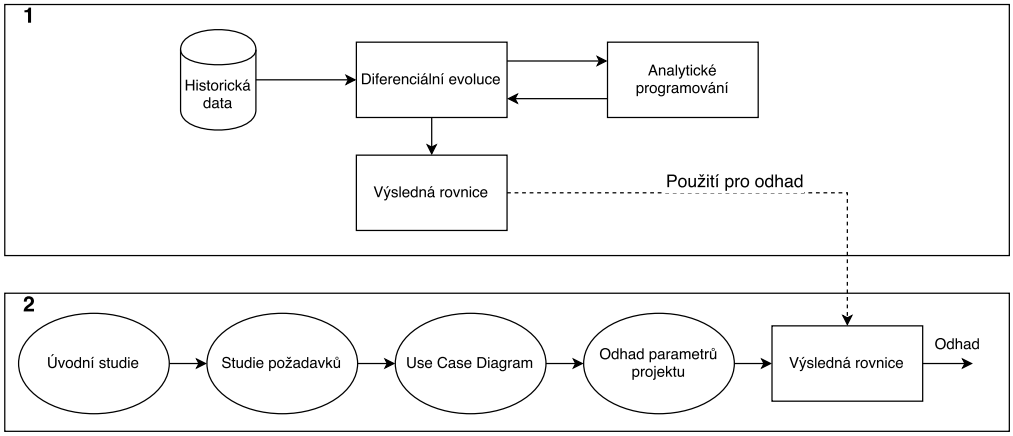
6.1 Postup návrhu nového frameworku odhadování

Nový framework je založen na úvodním stanovení základních charakteristik softwarového projektu pomocí techniky pro odhadování UCP, kdy byly využity parametry UUCW, UAW, TCF, ECF viz. kapitola 4.1. Analytické programování je použito pro syntézu nových matematických modelů z použitých datových sad, které budou dále podrobeny statistické analýze. Dále probíhal proces zkoumání vlastnosti této metody z hlediska její přesnosti na odhad a zjednodušování jejího použití. Finálním krokem bude porovnání této metody s dalšími interpretovatelnými statistickými metodami, které se v tomto oboru používají, např. lineární regrese či rozhodovací stromy.

6.2 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. Standardně by se pokračovalo výpočtem rovnice zmíněné v kapitole 4.1, tedy rovnicemi 4.5 a 4.6. Tímto by jsme získali odhad časového úsilí podle Karnerovi rovnice. V této práci je předpokládáno, že rovnice 4.5 a 4.6 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. Tato nová rovnice se může skládat z jakýchkoliv matematických i uživatelsky definovaných funkcí, a to včetně odhadu konstant. Podmínkou funkčnosti dané metody jsou historická

data projektů. Tato podmínka je však kladena také na ostatní metody, které spadají do kategorie metod řízených daty. Celý návrh řešení se zaměřuje na to, že každá společnost vyvíjející software má jiný model predikce časových náročností. Nicméně stejný princip může být aplikován také na odhad univerzálně platného nelineárního modelu. Záleží tedy pouze na vstupních, historických datech. Za předpokladu, že jsou vstupní data, data určité softwarové společnosti, bude model platný pouze pro tuto společnost. Pokud však bude algoritmu předložen větší vzorek dat z různých softwarových společností, je možné získat univerzálně platný model. Avšak toto platí jen za předpokladu, že budou tato data vhodně vybírána napříč softwarovými společnostmi např. pomocí stratifikovaného případně vícestupňového výběru dat.



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. Tento framework může být spuštěn na pozadí nezávisle na úkonech softwarového inženýra.

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.

6.3 Popis získaných dat a proměnných

Pro naplnění cílů výzkumu bylo nutné získat důvěryhodná data. Data byla získána především z publikací a výzkumu autorů věnujících se podobné problematice odhadování časových náročností softwarových produktů. Následuje výpis získaných datových sad.

- Dataset označen D1 je sloučením dvou datasetů :
  - Dataset z Technické univerzity v Poznani [67]
  - Dataset nacházející se v článku autora Subriady et al. [104]
- Dataset označen D2 získaný od autorů Šilhavý et al. [95, 96]

V prvních fázích výzkumu byly datasety používány zvlášť v pořadí v jakém docházelo k jejich získávání. Avšak za účelem porovnání prezentovaného frameworku byly dva menší datasety sloučeny a porovnávány s větším datasetem D2. Proměnných v datasetu je pět a jsou to UUCW, UAW, TCF, ECF a Effort [člověkohodiny]. Ze statistického hlediska jsou všechny proměnné kvantitativního, spojitého typu.

Dataset získaný z technické univerzity v Poznani od autora publikujícího v oboru softwarového inženýrství Ochodek et al. [67] sestává ze 14 softwarových projektů. Tyto softwarové projekty byly programovány v různých programovacích jazycích. Převládá zde však programovací jazyk JAVA. Co se týká rozsahu jednotlivých softwarových projektů, jedná se např. o CMS systémy, webové frontendy, bankovní systémy a podobné. Tedy jde o dataset, který se nezaměřuje pouze na webové aplikace nebo na určitý programovací jazyk.

Dataset získaný z článku autora Subriady et al. [104] sestává z 10-ti 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. [95, 96] 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.

6.4 Popis datové sady

Deskriptivní statistika

Tab. 6.1 Tabulka zobrazuje deskriptivní statistiku použitých datových sad

	n	$\bar{x}$	sd	med.	min	max	šik.	špič.
D1:UUCW	24	141,04	88,99	122,50	30,00	355,00	1,03	-0,01
D1:UAW	24	10,42	3,39	11,00	4,00	18,00	0,09	-0,58
D1:TCF	24	0,94	0,12	0,94	0,71	1,12	-0,17	-1,09
D1:ECF	24	0,88	0,14	0,90	0,51	1,19	-0,24	0,35
D1:Effort	24	1816,96	1111,98	1806,50	277,00	3950,00	0,32	-1,12
D2:UUCW	70	386,21	88,91	355,00	250,00	610,00	0,55	-0,75
D2:UAW	70	10,49	5,02	8,00	6,00	19,00	0,77	-1,31
D2:TCF	70	0,92	0,11	0,94	0,71	1,12	-0,26	-1,09
D2:ECF	70	0,86	0,12	0,88	0,51	1,08	-0,53	0,61
D2:Effort	70	6558,73	664,24	6406,00	5775,00	7970,00	0,55	-1,00

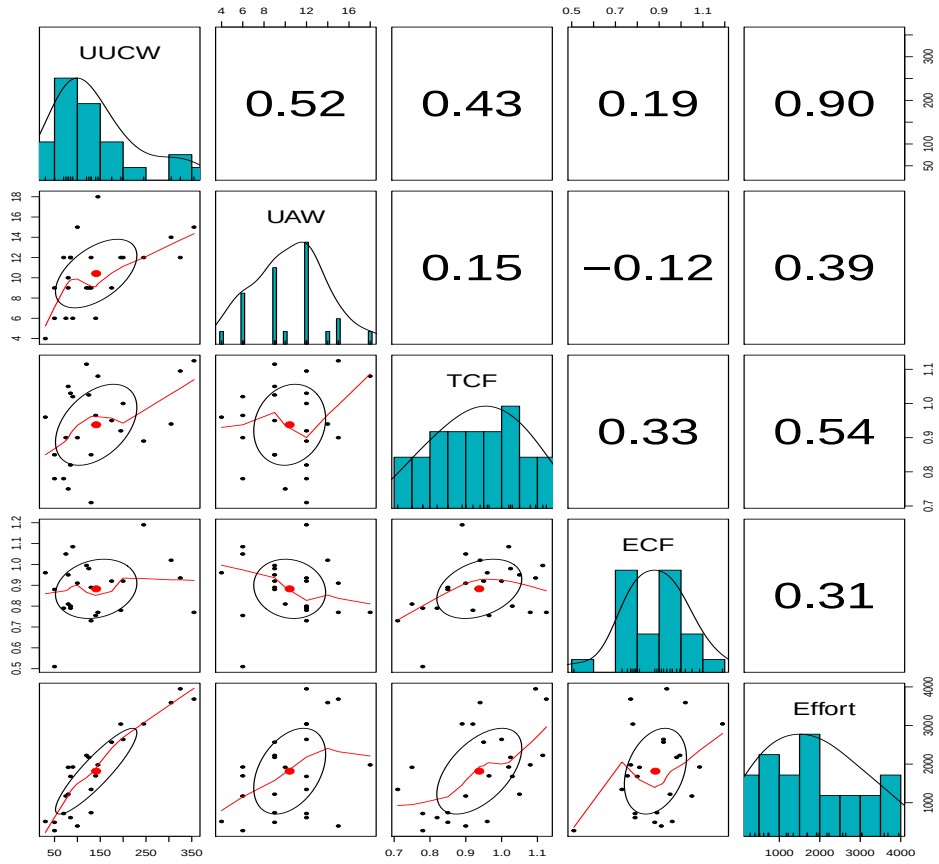
V tabulce 6.1 je výpočet deskriptivní statistiky pro obě použité datové sady. Z každé proměnné datasetu D1 je k dispozici 24 záznamů a to znamená, že tato datová sada neobsahuje chybějící hodnoty. V datasetu D2 je k dispozici 70 záznamů pro všechny proměnné, a také se tedy jedná o kompletní dataset. Dále je možné si povšimnout aritmetického průměru proměnné UUCW a Effort hlavně v porovnání obou datasetů. Je vidět, že UUCW pro dataset D1 dosahuje průměrné hodnoty 141 v porovnání s datasetem D2 386. Průměr proměnné UUCW u dataset D1 je 1816 člověkohodin a u D2 je to 6558 člověkohodin. Průměry ostatních

proměnných v porovnání obou datasetů jsou relativně shodné. Po revizi těchto hodnot se můžeme domnívat, že dataset D1 obsahuje převážně projekty menšího rozsahu. Dále je také zvláštní, že u datasetu D2 se průměr UUCW zvýšil o 2,7 krát, kdežto u proměnné Effort bylo zvýšení až 3,6 krát. Z tohoto pozorování si můžeme dělat představu, že výpočet Effort je závislý převážně na hodnotě UUCW. Co se týká výpočtu směrodatných odchylek jednotlivých proměnných (sd) ty jsou při porovnání datasetu převážně podobné až na proměnnou Effort. U datasetu D2 je až dvojnásobně nižší směrodatná odchylka. Jako vysvětlení se nabízí, že obsahuje více pozorování a tedy se rozptyl resp. směrodatná odchylka snižuje. Pro porovnání mediánů obou souborů napříč proměnnými můžeme tvrdit totéž co pro průměr, zase je dobré si povšimnout zejména mediánu proměnných UUCW a Effort. Co se týká minim a maxim, je nutné si povšimnout zejména minima u proměnné D1:UUCW, kde se nachází UUCW pouze 30, což samo o sobě byl buď velmi malý projekt nebo se jedná o extrémní hodnotu. Analýza šikmosti a špičatosti jednotlivých proměnných nevykazuje známky porušení normálního rozdělení hodnot. Normalita rozdělení jednotlivých proměnných bude dále analyzována vhodným statistickým testem.

Korelace

Na grafu 6.2 je vidět rozložení dat a Pearsonovi korelační koeficienty jednotlivých parametrů datasetu D1. Žádný z parametrů nevykazuje vizuální známky normálního rozdělení snad jen parametr ECF. Nejvíce nás však zajímá rozdělení závislé proměnné Effort. Tato proměnná vizuálně připomíná spíše lognormální rozdělení popř. jiné rozdělení zešikmené na levou stranu. Můžeme si také povšimnout korelačních koeficientů. Nejzajímavější je korelační koeficient mezi UUCW a Effort, který ukazuje na silnou kladnou korelaci. Podobná, avšak slabší je korelace mezi Effort a TCF. Mírně problematická je korelace mezi parametry projektu UAW a UUCW, která ukazuje na středně silnou kladnou korelaci. Problematické to je z důvodu, že v případě použití mnohonásobné lineární regrese bude docházet k multikolinearitě, což vede k nižší vypovídací hodnotě výsledného modelu.

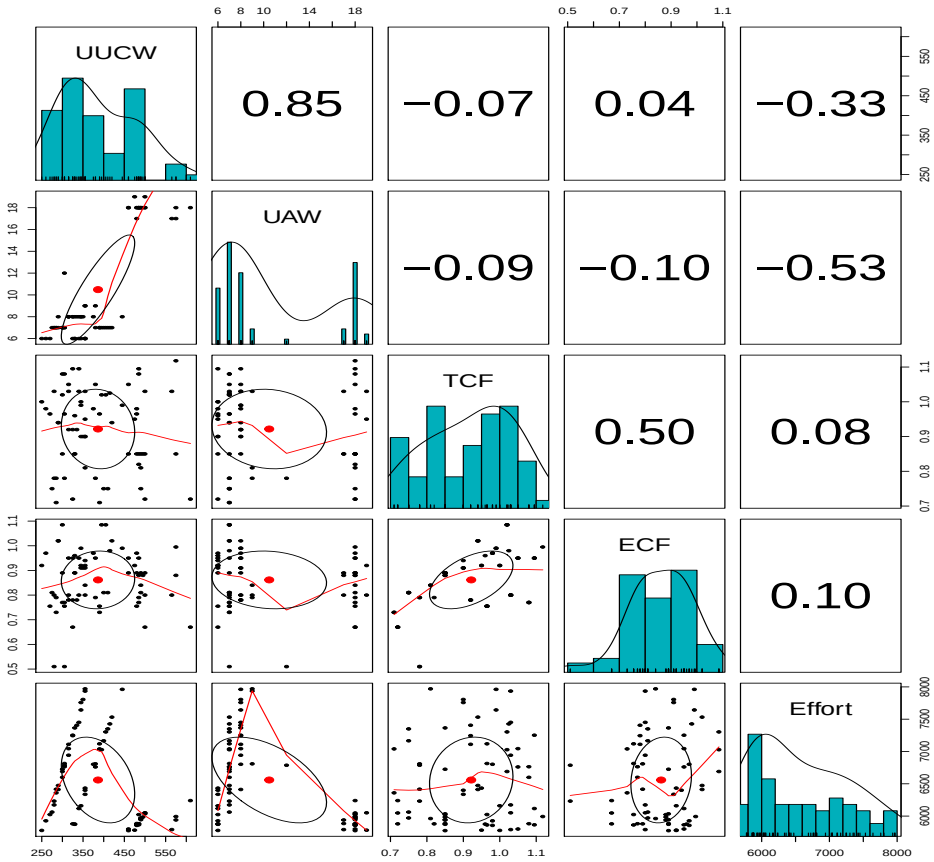
Na grafu 6.3 je vidět rozložení dat a Pearsonovi korelační koeficienty jednotlivých parametrů datasetu D2. Žádný z parametrů nevykazuje vizuální známky



Obr. 6.2 Histogram a korelační koeficienty jednotlivých parametrů datasetu D1

normálního rozdělení, zde ani parametr ECF. Nejvíce nás však zajímá rozdělení závislé proměnné Effort. Tato proměnná vizuálně připomíná spíše lognormální rozdělení. Můžeme si také povšimnout korelačních koeficientů. Nejzajímavější je korelační koeficient mezi UUCW, UAW a Effort, které ukazují na zápornou korelaci. Mírně problematická je korelace mezi parametry projektu UAW a UUCW, která ukazuje na silnou kladnou korelaci. Problematické to je z důvodu, že v případě použití mnohonásobné lineární regrese bude docházet k multikolinearitě, což vede k nižší vypovídací hodnotě výsledného modelu.

Při porovnání korelací obou datasetů jsou patrné jisté odlišnosti v obou datasetech a jak jsou jednotlivé parametry korelovány s nezávislou proměnnou Effort.



Obr. 6.3 Histogram a korelační koeficienty jednotlivých parametrů datasetu D2

Můžeme si povšimnout zejména kladné korelace parametrů UUCW a UAW s parametrem Effort u D1 a záporných korelací u stejných proměnných u datasetu D2. Silnější korelace mezi parametrem UUCW a UAW je totožná u obou datasetů.

Testy normality

Pro ověření normality jednotlivých proměnných byl použit Shapiro-Wilk test.

- H_0 : Data pocházejí z normálního rozdělení

- H_A : Data nepocházejí z normálního rozdělení

Tab. 6.2 Tabulka zobrazuje test Shapiro-Wilk pro proměnné obou datasetů

	W	p-hodnota	Hypotéza
D1:UUCW	0,8749	0,0066	H_A
D1:UAW	0,9474	0,2383	H_0
D1:TCF	0,9706	0,683	H_0
D1:ECF	0,9655	0,5577	H_0
D1:Effort	0,9445	0,2052	H_0
D2:UUCW	0,9306	8e-04	H_A
D2:UAW	0,7115	0	H_A
D2:TCF	0,9515	0,0087	H_A
D2:ECF	0,9589	0,0218	H_A
D2:Effort	0,901	0	H_A

V tabulce 6.2 je výpočet Shapiro-Wilk testů ověření normality pro proměnné obou datových sad. Je možné si povšimnout rozdílů mezi datasety. Pro dataset D1 se většina parametrů, kromě UUCW, jeví jako normálně rozdělena. Avšak pro dataset D2, se podařilo nasbírat dostatek důkazů pro zamítnutí nulové hypotézy na hladině významnosti 5 %. Dále v práci tedy budeme předpokládat, že pro dataset D2 žádná proměnná nenásleduje normální rozdělení dat. Je velmi překvapivé, že pro oba datasety vyšly, tak diametrálně rozdílné výsledky. Důvod by jsme mohly najít v množství pozorování v jednotlivých datasetech.

6.5 Návrh experimentů

Pro splnění cílů disertační práce byly navrhnuty následující experimenty, které budou provedeny pro oba datasety, aby tak mohla být ověřena validita použitých metod.

Tab. 6.3 Tabulka nastavení algoritmu analytického programování

GFS2	plus, mínus, násobení, dělení, umocnění
GFS1	log , ln , odmocnina , absolutní hodnota, sinus, kosinus
GFS0	numerická konstanta, UUCW, UAW, TCF, ECF

V tabulce 6.3 je zaznamenán seznam funkcí pro analytické programování. Jak je patrné z tabulky byly použity standardní matematické funkce, žádné funkce uživatelsky definované. Do symbolické regrese byly zařazeny i funkce nelineární, jako je například logaritmus nebo harmonické funkce sinus a kosinus. V množině GFS0 jsou zaznamenány použité terminální symboly, jedná se o parametry projektu metody UCP a numerická konstanta.

Tab. 6.4 Tabulka nastavení algoritmu diferenciální evoluce

Parametr	Hodnota
NP	100
CR	0,8
F	0,3
Gen	500
Specimen	13

V tabulce 6.4 je zaznamenáno nastavení algoritmu diferenciální evoluce. V prvních fázích práce byly tyto hodnoty nastaveny dle doporučení literatury. Později se experimentálně ukázalo toto nastavení jako vhodnější. Hlavně parametr Gen může být vnímán jako vysoká hodnota, ale pro účely statistického testování bylo nutné získat výsledky s co nejmenším rozptylem. Čím vyšší bude tento parametr, tím větší je pravděpodobnost, že skončíme v minimu i když třeba lokálním. Maximální počet elementů vygenerované rovnice byl nastaven na 13. Důvodem zvolení právě této hodnoty je možné porovnání s metodou mnohonásobné lineární regrese. Mnohonásobná lineární regrese by v tomto případě obsahovala třináct symbolů viz. rovnice 6.1 bez systematické chyby ϵ. Tedy devět terminálních symbolů $\{\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, UUCW, UAW, TCF, ECF\}$ a čtyři neterminální symboly matematické funkce sčítání.

$$\hat{y} = \beta_0 + \beta_1 UUCW + \beta_2 UAW + \beta_3 TCF + \beta_4 ECF + \epsilon \tag{6.1}$$

Pro naprogramování jádra prezentovaného frameworku byl použit skriptovací jazyk LUA 5.1.5. Pro deskriptivní a inferenční statistické výpočty byl použit

statistický software R ve verzi 3.6.1. Pro výpočty síly testu a počtu vzorků pro jednotlivé experimenty byla použita aplikace G*Power [28].

6.5.1 Výzkumná Otázka 1

První výzkumná otázka se věnuje lineární závislosti mezi ohodnocením softwarového projektu metodou UCP a skutečnou časovou náročností softwarového projektu.

Motivací pro tuto výzkumnou otázku je hypotéza, že by jsme neměli apriori předpokládat lineární závislost mezi hodnocením softwarového projektu a mezi skutečnou časovou náročností.

Pro tento experiment bude sestaven model viz. rovnice 6.1. Závislá proměnná bude skutečná časová náročnost (Effort), nezávislé proměnné jsou UUCW, UAW, TCF, ECF. Dále budou využity vlastnosti statistické metody mnohonásobné lineární regrese. Aby mohla být lineární regrese použita je zapotřebí splnění mnoha podmínek. Jednou z významnějších je, že rezidua musí být normálně rozdělena.

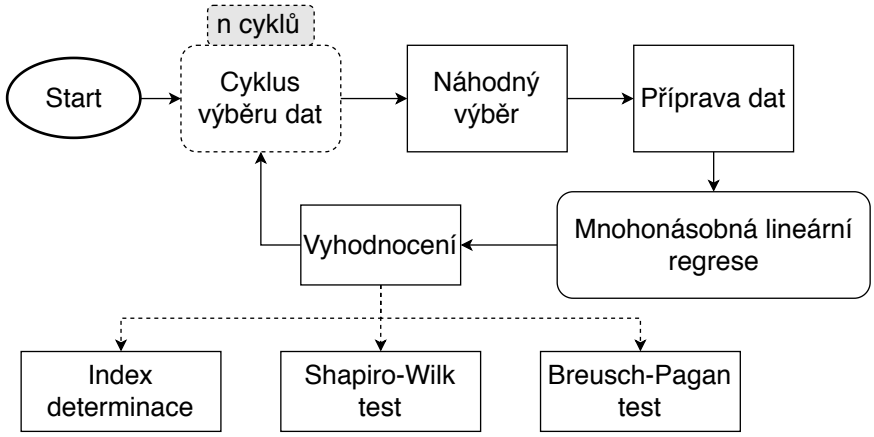
Pro ověření podmínek bude využit test normality dat Shapiro-Wilk test na rezidua a pro ověření rozptylu na reziduích bude využit Breusch-Pagan test. Dále bude využita metrika indexu determinace R^2, která slouží primárně k odhadu vhodnosti modelu na základě vysvětleného rozptylu.

Průběh experimentu

Data budou rozdělena s využitím metody náhodného výběru s opakováním. Budou využity klasické hodnoty tedy 66 % dat bude použito na trénovaní modelu, zbytek pak na testování. Každá sada bude podrobena statistickému šetření popsaného níže. Návrh experimentu je naznačen na obrázku 6.4.

V první fázi experimentu bude vypočteno 10 průběhů, které budou sloužit jako pilotní průzkum. Na základě rozptylů bude určen počet vzorků experimentu s využitím výpočtů statistické síly. Statistická síla $1 - \beta$ bude nastavena na klasickou hodnotu $1 - \beta = 0,8$.

Ve druhé fázi experimentu budou ověřovány následující tři hypotézy :



Obr. 6.4 Diagram experimentu pro zjištění výzkumné otázky 1

- H1 Méně než v 50ti % případů jsou rezidua normálně rozdělena
- H2 Méně než 50 % případů mají rezidua konstantní rozptyl
- H3 Průměrný index determinace je nižší než 0,6

Hodnota indexu determinace 0,6 je standardní doporučovaná hodnota pro model, na jehož základě se budou pořizovat odhady. Hypotéza H1 a H2 bude otestována pomocí testu o proporcích. Hypotéza H3 bude otestována pomocí jednovýběrového t-testu.

Třetí fáze experimentu bude vyhodnocení, interpretace, závěr a porovnání obou datových sad.

6.5.2 Výzkumná Otázka 2

Druhá výzkumná otázka se věnuje parametrům metody UCP a jejich vlivu na výsledný odhad resp., které parametry projektu metody UCP mají největší vliv na přesnější odhad.

Motivací pro tento experiment je hypotéza, že ne všechny parametry projektu metody UCP přispívají k přesnějším odhadům stejnou mírou. Analýzou této výzkumné otázky může dojít k tomu, že některé parametry projektu ohodnocené

metodou UCP nemají pro výsledný odhad opodstatnění a může dojít ke zjednodušení odhadovacího modelu resp., může dojít ke zmenšení prohledávaného prostoru, a tím k rychlejšímu průběhu vytváření modelů. Druhým důvodem je zjištění konzistence mezi datasety, tedy zda je výběr parametrů konzistentní napříč datasety. Tato výzkumná otázka bude řešena pomocí prezentovaného frameworku tedy pomocí diferenciální evoluce a analytického programování.

Průběh experimentu

V první fázi experimentu budou data rozdělena s využitím metody náhodného výběru s opakováním. Budou využity klasické hodnoty tedy 66 % dat bude použito na trénovaní modelu, zbytek pak na testování. Každá sada bude podrobena statistickému šetření popsaného níže.

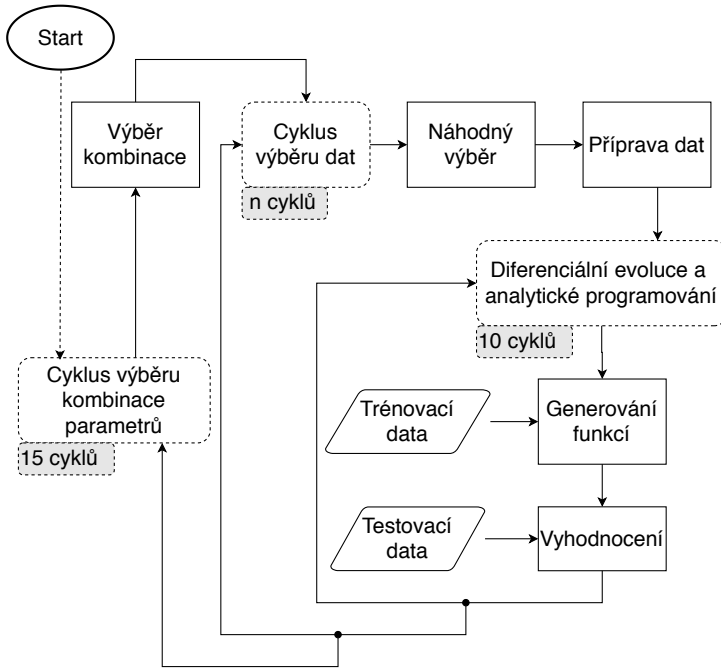
Pro vyřešení této otázky bude sestaven následující experiment viz. obrázek 6.5 . Nejprve bude vytvořena tabulka všech kombinací parametrů projektů, viz tabulka 6.5 projektu UUCW, UAW, TCF a ECF. Následně bude pro každou kombinaci vygenerováno 10 rovnic. A bude vypočtena hodnota MMRE. Tyto hodnoty MMRE budou dále podrobeny výpočtu statistické síly, aby mohl být určen počet celkových generovaných rovnic. Statistická síla $1-\beta$ bude nastavena na klasickou hodnotu $1-\beta = 0,8$.

Tab. 6.5 Tabulka zobrazuje kombinace parametrů

Jeden par.	Dva par.	Tři par.	Čtyři par.
UUCW	UUCW,UAW	UUCW,UAW,TCF	UUCW,UAW,TCF,ECF
UAW	UUCW,TCF	UUCW,TCF,ECF	
TCF	UUCW,ECF	UUCW,UAW,ECF	
ECF	UAW,TCF	UAW,TCF,ECF	
	UAW,ECF		
	TCF,ECF		

Ve druhé fázi experimentu bude ověřována následující hypotéza :

- H1 Všechny kombinace parametrů projektu metody UCP mají stejný průměr/medián MMRE



Obr. 6.5 Diagram experimentu pro zjištění výzkumné otázky 2

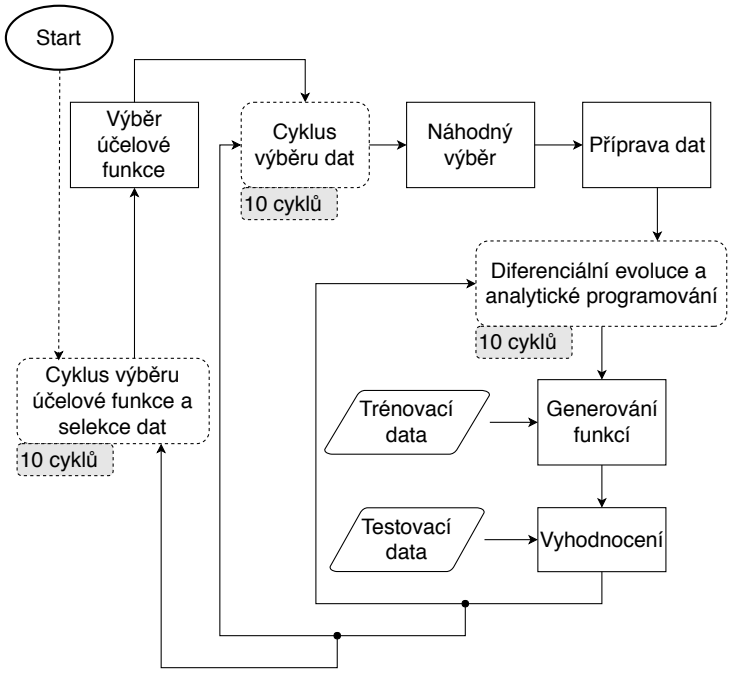
Výsledky jednotlivých MMRE budou nejprve přezkoumány na normální rozdělení testem Shapiro-Wilk. Poté bude vybrán vhodný test pro ověření hypotézy. V případě ověření normality bude použita analýza rozptylu (ANOVA), pokud se normální rozdělení MMRE nepodaří ověřit, bude použita metoda Kruskal-Wallis (neparametrická ANOVA).

Třetí fáze experimentu bude vyhodnocení, interpretace, závěr a porovnání obou datových sad.

6.5.3 Výzkumná Otázka 3

Třetí výzkumná otázka se věnuje vlivu použité účelové funkce na přesnost výsledného odhadu.

Motivací pro tento experiment je hypotéza, že ne všechny účelové funkce přispívají k přesnějším odhadům stejnou mírou. Analýzou této výzkumné otázky dojde k tomu, že bude určena účelová funkce, která podá přesnější odhady. Tato



Obr. 6.6 Diagram experimentu pro zjištění vhodnosti účelových funkcí

výzkumná otázka bude řešena pomocí prezentovaného frameworku tedy pomocí diferenciální evoluce a analytického programování.

Průběh experimentu

V první fázi experimentu budou data rozdělena s využitím metody náhodného výběru s opakováním. Budou využity klasické hodnoty tedy 66 % dat bude použito na trénovaní modelu, zbytek pak na testování. Každá sada bude podrobena statistickému šetření popsaného níže.

Byl navržen experiment na obrázku 6.6. Pro každou ze 3 účelových funkcí (MMRE, MSE, LAD, definovaných v kapitole 4.5). Aby byla zachována statistická síla testu, proběhne výpočet počtu vzorků každé účelové funkce.

Ve druhé fázi experimentu bude ověřována následující hypotéza :

- H1 Všechny účelové funkce mají stejný průměr/medián MMRE

Výsledky jednotlivých MMRE budou nejprve přezkoumány na normální rozdělení testem Shapiro-Wilk. Poté bude vybrán vhodný test pro ověření hypotézy. V případě ověření normality bude použita analýza rozptylu (ANOVA), pokud se normální rozdělení MMRE nepodaří ověřit, bude použita metoda Kruskal-Wallis (neparametrická ANOVA).

Třetí fáze experimentu bude vyhodnocení, interpretace, závěr a porovnání obou datových sad.

6.5.4 Výzkumná Otázka 4

Čtvrtá výzkumná otázka se věnuje porovnání frameworku s odhadem podle standardní rovnice metody UCP.

Motivací pro tento experiment je zjištění, zda nový framework poskytuje přesnější odhady než standardní rovnice metody UCP a zda je tento rozdíl statisticky významný.

Průběh experimentu

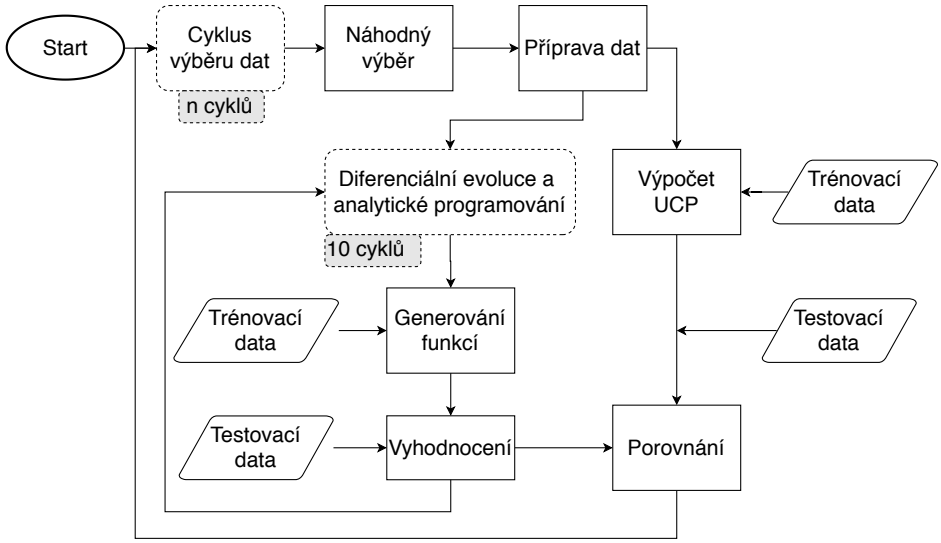
V první fázi experimentu budou data rozdělena s využitím metody náhodného výběru s opakováním. Budou využity klasické hodnoty tedy 66 % dat bude použito na trénovaní modelu, zbytek pak na testování. Každá sada bude podrobena statistickému šetření popsaného výše.

Byl navržen experiment na obrázku 6.7. Pro pilotních 10 výběrů budou sestaveny dva modely jeden podle standardní rovnice UCP a druhý bude vygenerován novým frameworkem. Nad těmito modely bude vypočtena míra kritéria kvality MMRE. Ten bude dále statisticky analyzován, aby mohlo dojít k výpočtu počtu vzorků. Statistická síla $1 - \beta$ bude nastavena na klasickou hodnotu $1 - \beta = 0,8$.

Ve druhé fázi experimentu bude ověřována následující hypotéza :

- H1 Průměr MMRE obou modelů je stejný

Výsledky jednotlivých modelů budou nejprve přezkoumány na normální rozdělení testem Shapiro-Wilk. Poté bude vybrán vhodný test pro ověření hypotézy. V případě ověření normality bude použit párový dvouvýběrový t-test o shodnosti



Obr. 6.7 Diagram experimentu pro zjištění výzkumné otázky 4

průměrů, pokud se normální rozdělení nepodaří ověřit, bude použit neparametrický párový Wilcoxonův test.

Třetí fáze experimentu bude vyhodnocení, interpretace, závěr a porovnání obou datových sad.

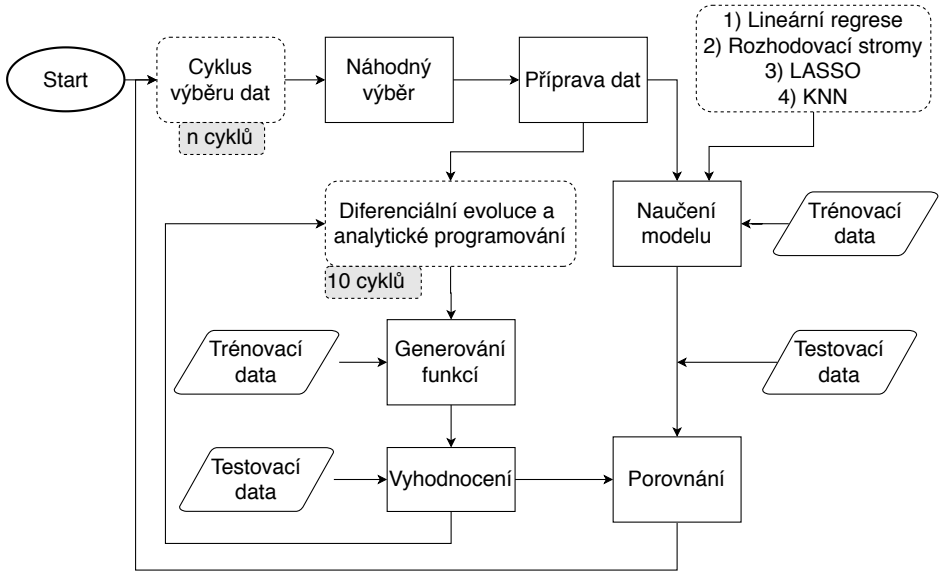
6.5.5 Výzkumná Otázka 5

Pátá výzkumná otázka se věnuje porovnání frameworku s odhadem podle jiných modelů jako je lineární regrese, neuronová síť nebo rozhodovací stromy.

Motivací pro tento experiment je zjištění, zda nový framework poskytuje přesnější odhady a zda je tento rozdíl statisticky významný, případně zda se nový framework vyrovná standardně používaným modelům.

Průběh experimentu

V první fázi experimentu budou data rozdělena s využitím metody náhodného výběru s opakováním. Budou využity klasické hodnoty tedy 66 % dat bude použito na trénovaní modelu, zbytek pak na testování. Každá sada bude podrobena statistickému šetření popsaného níže.



Obr. 6.8 Diagram experimentu pro zjištění výzkumné otázky 5

Jednotlivé modely jsou vybrány tak, aby zahrnovaly celou škálu různých funkčních předpisů a struktur. Ať se jedná o předpoklad linearity, klastrovací algoritmy nebo rozhodování formu if-else. Všechny použité algoritmy byly použity jako knihovny statistického softwaru R.

Byl navržen experiment na obrázku 6.8. Pro pilotních 10 výběrů budou sestaveny následující modely :

- Mnohonásobná lineární regrese (lm) [15, 117]
- Kroková lineární regrese (leapSeq) [35]
- Neuronová síť (neuralnet) [83, 84, 4]

- Ridge regrese (ridge) [33]
- LASSO regrese (lasso) [125]
- Rozhodovací strom CART (rpart) [10]
- Obecný lineární model (glm) [124, 24, 114]
- K-nearest neighbors (knn) [85, 115]
- Support vector machine (svm) [16, 27]

V závorce je uveden název knihovny, která byla pro daný model ve statistickém softwaru R využívaná. Žádný z modelů nebyl optimalizován a pokud měl daný model parametry, tak byly nastaveny výchozí, doporučované hodnoty. Nad těmito modely budou vypočítány míry kritéria kvality MMRE. Tento bude dále statisticky analyzován, aby mohlo dojít k výpočtu počtu vzorků. Statistická síla $1 - \beta$ bude nastavena na klasickou hodnotu $1 - \beta = 0,8$.

Ve druhé fázi experimentu bude ověřována následující hypotéza :

- H1 Průměr MMRE je u všech modelů totožný

Výsledky jednotlivých modelů budou nejprve přezkoumány na normální rozdělení testem Shapiro-Wilk. Poté bude vybrán vhodný test pro ověření hypotézy. V případě ověření normality bude použit ANOVA pro opakované měření, pokud se normální rozdělení nepodaří ověřit, bude použit neparametrický Friedmanův test.

Třetí fáze experimentu bude vyhodnocení, interpretace, závěr a porovnání obou datových sad.

7 Výsledky experimentů a jejich interpretace

Během práce na této disertační práci bylo dosaženo několika významných výsledků, které byly publikovány v odborných časopisech a na odborných konferencích.

7.1 Výzkumná Otázka 1

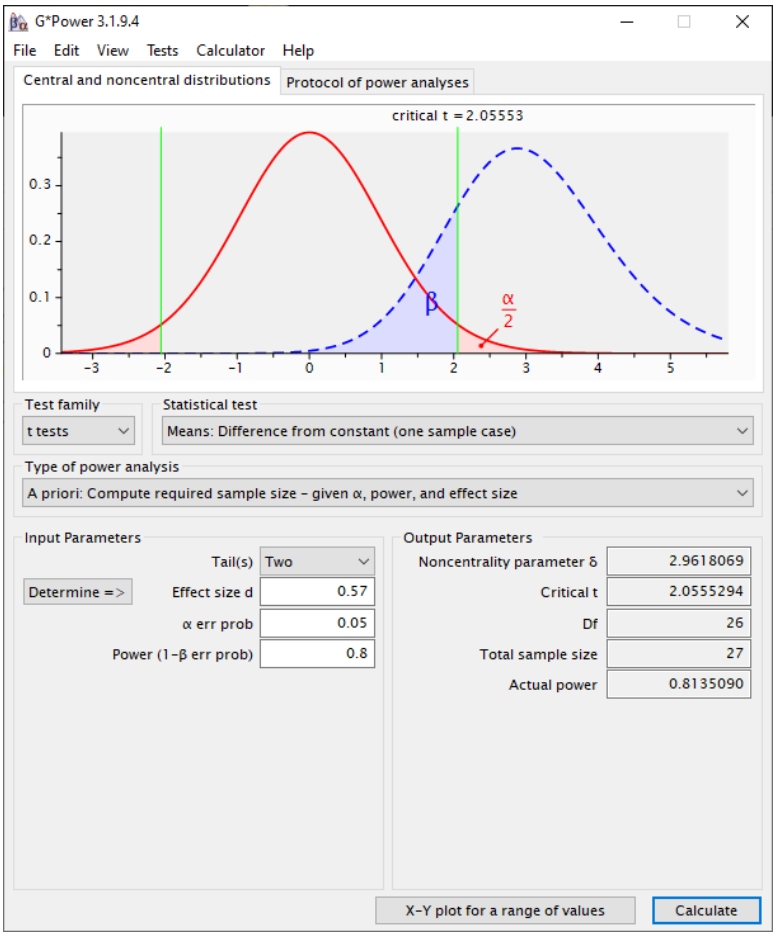
V první fázi experimentu bylo vypočteno 10 průběhů, které byly použity pro výpočet odhadu počtu celkových vzorků.

Tab. 7.1 Výpočet průměru jednotlivých znaků

Dataset	R^2	Shapiro-Wilk	Breusch-Pagan
D1	0,81	0,97	1,00
D2	0,29	0,80	1,00

V tabulce 7.1 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 97 % cyklů u D2 je to 80 % cyklů. U obou datasetů z těchto 10 cyklů všechny průchody nezaznamenaly splnění homoskedasticity reziduí [106]. Je také vidět, že index determinace je nejvíc variabilním faktorem pro oba datasety, a proto byl použit pro výpočet velikosti vzorků, tak aby byla zachována síla testu $1 - \beta > 80\%$. Z výpočtu síly t-testu pro průměr je zapotřebí 27 vzorků aby byla zachována síla testu. Při této velikosti vzorků bude síla testu $1 - \beta > 81,18\%$. Výpočet byl proveden v programu G*Power, jak je vidět na obrázku 7.1. Dále tedy bylo vygenerováno těchto 27 vzorků pro každý dataset a bylo přistoupeno k testování hypotéz.

Z tabulky 7.2 je vidět, že průměrný index determinace pro dataset D1 je 0,66 a pro D2 je to 0,3. Což lze interpretovat tak, že lineární regresní model vysvětlí průměrně 66 % variability resp. 30 % variability . Což je velmi nízká hodnota pro odhadovací model. Relativní četnosti zamítnutí nulové hypotézy Shapiro-wilk testu a Breusch-Pagan testu jsou velmi nízké. To znamená, že rezidua jsou v mnoha případech normálně rozdělena. Avšak můžeme si povšimnout



Obr. 7.1 Výpočet počtu vzorků pro výzkumnou otázku 1, tak aby síla testu byla $1 - \beta > 80\%$

Tab. 7.2 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

poklesu relativní četnosti zamítnutí nulové hypotézy o homoskedasticitě reziduí pro dataset D2.

Tab. 7.3 Výpočet hypotéz

Dataset	Statistika	Stupňů volnosti	p-hodnota
D1	$t = -13,83$	26	1
D1	$\chi^2 = 21,333$	1	1
D1	$\chi^2 = 3,7037$	1	0,97
D2	$t = -26,123$	26	$< 0,01$
D2	$\chi^2 = 14,815$	1	1
D2	$\chi^2 = 7,2593$	1	0,99

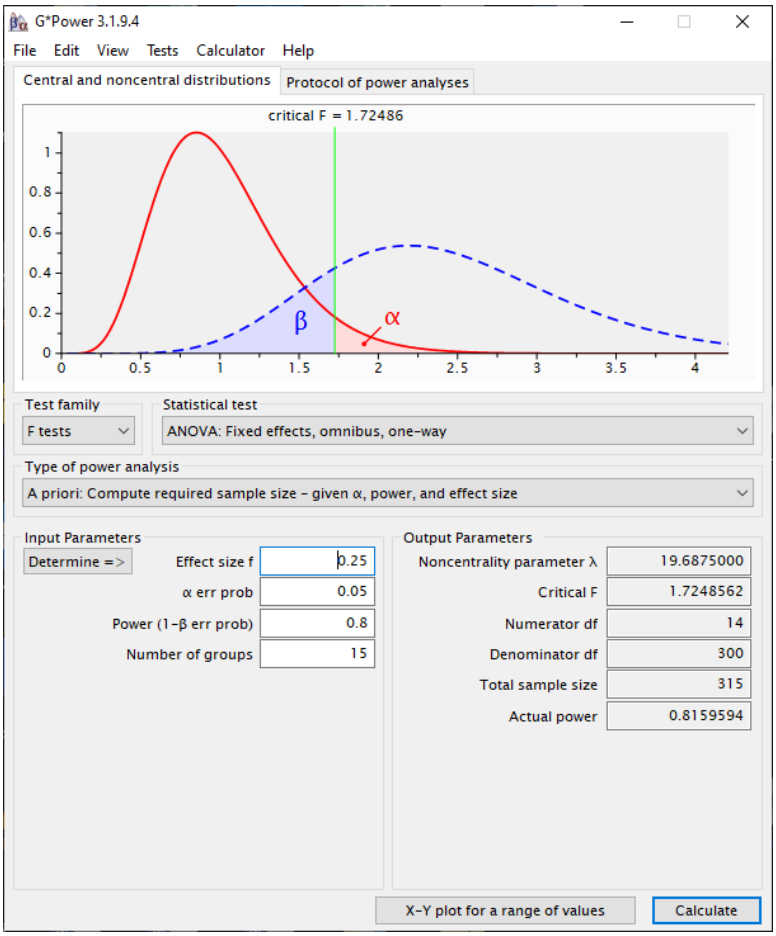
V tabulce 7.3 je vidět, že p-hodnota je pod hodnotou 0,05 jen u indexu determinace pro dataset D2. Máme tedy dostatek důkazů, že index determinace je menší než 0,6. Dále také budeme předpokládat, že rezidua s 95% spolehlivostí s více jak 50 % případů normálně rozdělena, a že rozptyl reziduí jsou s 95% spolehlivostí s více jak 50 % případů konstantní.

7.1.1 Shrnutí výsledků výzkumné otázky 1

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. Že na tato vybraná data nemohla být použita přinejmenším metoda nejmenších čtverců (standardní odhadovací metoda mnohonásobné lineární regrese), jelikož by odhad byl statisticky nekonzistentní. Docházelo by tedy k systematickému podhodnocování nebo nadhodnocování odhadu.

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.

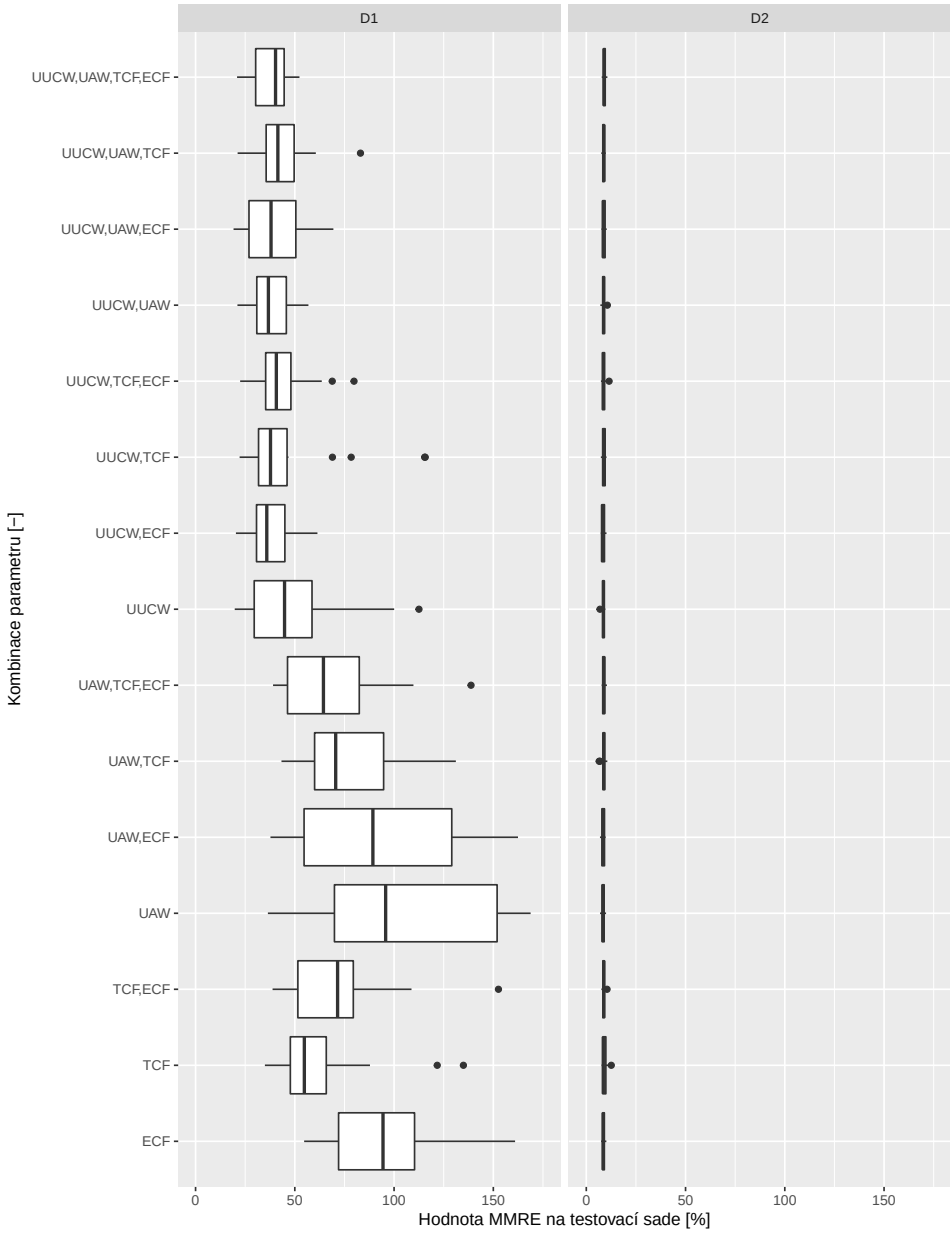


Obr. 7.2 Výpočet počtu vzorků pro výzkumnou otázku 2, tak aby síla testu byla $1 - \beta > 80\%$

7.2 Výzkumná Otázka 2

V první fázi ověření této výzkumné otázky byl proveden výpočet počtu vzorků, tak aby po použití testu ANOVA byla zachována statistická síla $1 - \beta > 0,8$. Po výpočtu je nutné získat dohromady 315 vzorků. Jelikož máme 15 skupin pro porovnání je potřeba získat z každé skupiny (kombinace parametrů) 21 vzorků. Výpočet byl proveden v programu G*Power, jak je vidět na obrázku 7.2.

Výsledky tohoto experimentu jsou znázorněny na grafu 7.3. Na datasetu D1 si můžeme povšimnout dvou důležitých zjištění. První rovnice, které měly přístup



Obr. 7.3 Statistické porovnání jednotlivých skupin parametrů projektů

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 cca 30 % MMRE. Co se týká datasetu D2, zde si můžeme povšimnout, že všechny kombinace parametrů si vedou velmi podobně (dále v textu bude ověřeno statistickým testem). Průměrná relativní chyba na testovacích datech datasetu D2 je 8,7 %.

Graf datasetu D2 je dále znázorněn na obrázku 7.4, jelikož graf 7.3 měl zobrazit porovnání výsledků mezi datasety. 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.

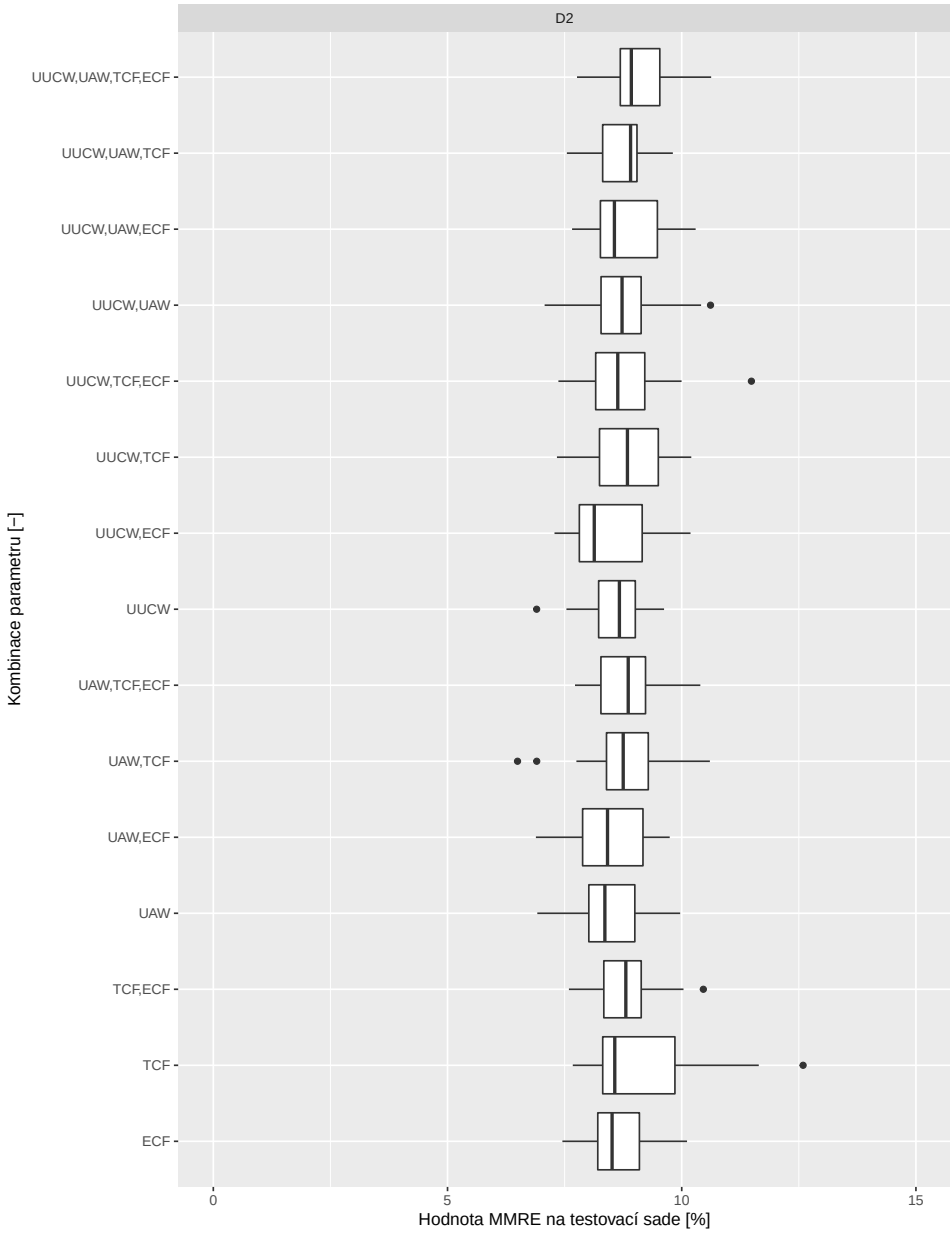
7.2.1 Statistické ověření

Tab. 7.4 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 7.4 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 datasetu 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.

V tabulce 7.5 je znázorněno zařazení do skupin Tukey post-hoc analýzy. Skupiny jsou vytvořeny na základě toho, zda vzájemné a vícenásobné porovnání dosažených výsledků jednotlivých metod pochází, na hladině významnosti 5 %, ze stejné populace. Zde je možné si povšimnout hlavně skupiny "f", která se vyskytuje jen v přítomnosti parametru UUCW. Dále je možné si povšimnout



Obr. 7.4 Statistické porovnání jednotlivých skupin parametrů projektů

skupiny "df". Tyto dvě kombinace parametrů mají stejný průměr, avšak jedna z nich obsahuje parametr ECF, kromě UUCW a TCF. To může ukazovat na

Tab. 7.5 Post-hoc analýza ověření diferencí u datasetu D1

Parametr	Skupina
ECF	b
TCF	cd
TCF,ECF	bde
UAW	a
UAW,ECF	bc
UAW,TCF	bc
UAW,TCF,ECF	bd
UUCW	df
UUCW,ECF	f
UUCW,TCF	df
UUCW,TCF,ECF	df
UUCW,UAW	f
UUCW,UAW,ECF	f
UUCW,UAW,TCF	ef
UUCW,UAW,TCF,ECF	f

nevýznamnost použití parametru ECF. ECF má pak samostatnou skupinu "b", která se poté objevuje v kombinaci s UAW i TCF. Velmi zajímavé je, že samotný parametr UAW je ve skupině "a", která není ve skupině s žádným jiným parametrem a tedy je statisticky významně rozdílná od ostatním parametrů. Bohužel parametr samotný parametr UAW dosahoval nejhorších výsledků, jak je vidět na grafu 7.3.

7.2.2 Shrnutí výsledků výzkumné otázky 2

Hlavní motivací, řešení této výzkumné otázky je fakt, že prohledávání prostoru o takové dimensionalitě je výpočetně velmi náročné. Snížení počtu dimenzí má na velikost prohledávaného prostoru velký vliv. Cílem tedy bylo dokázat, že je možné snížení dimenze snížením počtu vstupních parametrů. Cílem také bylo zjistit, zda je některý s parametrů projektu nutné zahrnout do GFS.

Pro menší dataset D1 je zjištění z grafu vypovídající. Jestliže bylo povoleno použití parametru UUCW, tak framework dosahoval statisticky významně lep-

ších 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.

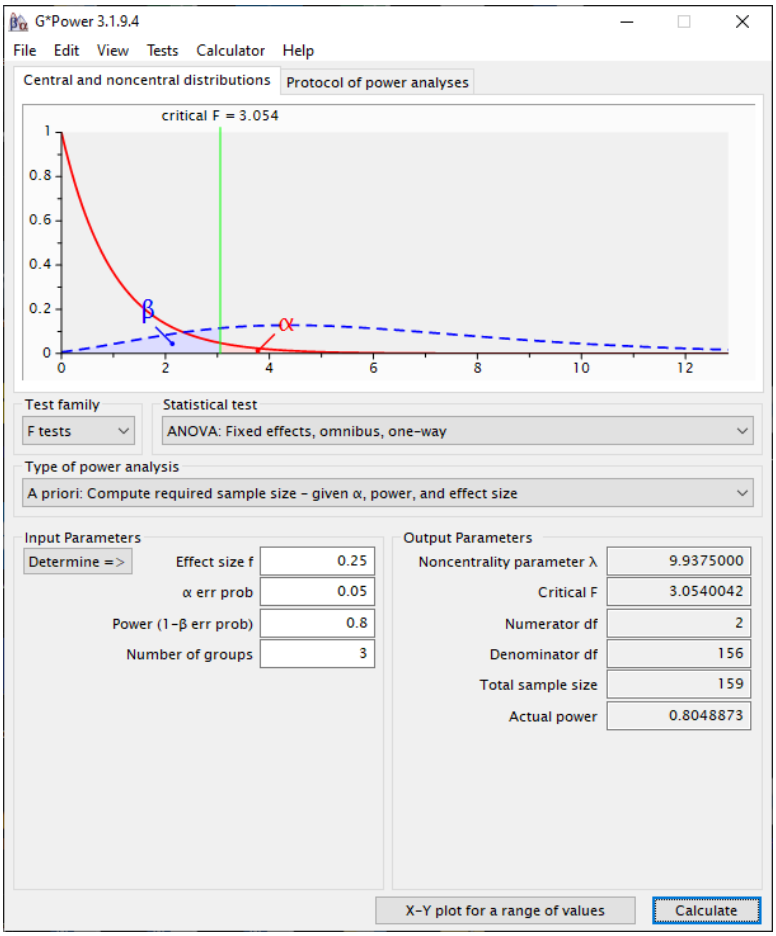
Možných faktorů, proč u datasetu D1 je možné vynechat tři parametry projektu (UAW,TCF a ECF) a u D2 ne, je několik. Zejména jde o velikost obou datasetů, kde dataset D1 je podstatně menší, a tedy máme větší volnost v generování funkcí. Čím více máme projektů na základě nichž dochází ke generování rovnic tím menší jsou naše možnosti a lokálních minim na prohledávané funkci ubývá. Dalším možným faktorem je variabilita jednotlivých datasetů, což souvisí i s výběrem softwarových projektů v jednotlivých datasetech.

Z výsledků tedy vyplývá, že snížení výpočetní náročnosti pomocí redukce vstupních parametrů není možné. Je tedy vhodné, aby všechny parametry byly frameworku k dispozici.

7.3 Výzkumná Otázka 3

V první fázi ověření této výzkumné otázky byl proveden výpočet počtu vzorků, tak aby po použití testu ANOVA byla zachována statistická síla $1 - \beta > 0,8$. Po výpočtu je nutné získat dohromady 159 vzorků. Jelikož máme 3 skupiny (MSE, LAD, MMRE) pro porovnání je potřeba získat z každé skupiny 53 vzorků. Výpočet byl proveden v programu G*Power, jak je vidět na obrázku 7.5.

Výsledky tohoto experimentu jsou znázorněny na grafu 7.6. Na datasetu D1 si můžeme povšimnout, že všechny 3 použité účelové funkce mají podobný medián a navzájem se překrývají, dalo by se tedy usuzovat, že zde nebude zaznamenán statisticky významný rozdíl. 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



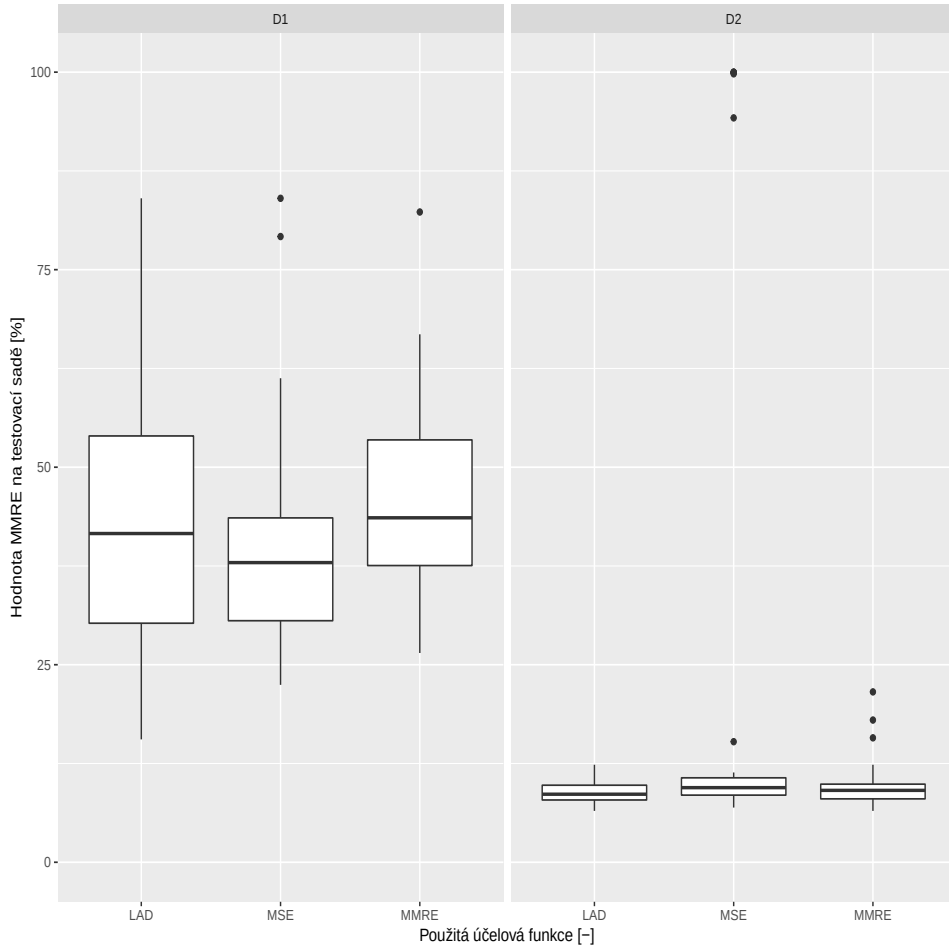
Obr. 7.5 Výpočet počtu vzorků pro výzkumnou otázku 2, tak aby síla testu byla $1 - \beta > 80\%$

extrémních hodnot. Účelové funkce MMRE a LAD si vedla konzistentně podobně na obou datasetech.

7.3.1 Statistické ověření

Vzhledem k výsledkům MSE u datasetu D2 bude použita neparametrická verze testu ANOVA a tou je test Kruskal-Wallis.

V tabulce 7.6 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



Obr. 7.6 Statistické porovnání jednotlivých skupin použitých účelových funkcí

Tab. 7.6 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

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.

Tab. 7.7 Matice Dunn-Bonferroni post hoc test pro dataset D1

	LAD	MMRE
MMRE	0,1234	
MSE	0,0506	0,0026

V tabulce 7.7 je vidět, že pouze p-hodnota pro porovnání MSE a MMRE je menší než 0,05. Účelové funkce MSE a LAD nemají statisticky významně odlišný medián hodnoty MMRE na testovacím datasetu.

Tab. 7.8 Matice Dunn-Bonferroni post hoc test pro dataset D2

	LAD	MMRE
MMRE	0,2866	
MSE	0,0033	0,0157

V tabulce 7.8 je vidět, že p-hodnota pro porovnání MSE-MMRE a MSE-LAD na datasetu D2 je menší než 0,05. Účelová funkce MSE má tedy na testovacích datech statisticky významně rozdílný medián MMRE.

7.3.2 Shrnutí výsledků výzkumné otázky 3

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. Toto se ukázalo u použití účelové funkce MSE pro druhý dataset D2. Nutno podotknout, že při výpočtu se změnil pouze dataset, funkce byla použita stejná jak pro D1, tak pro D2. Při výpočtu tedy chyba nemohla nastat a na datasetu D2 byly pro funkci MSE naměřeny extrémní hodnoty, které dále znemožňují použití této

účelové funkce. Důvodem pro naměření takto extrémních hodnot je zřejmě ve způsobu výpočtu, kdy se minimalizuje druhá mocnina chyby. Tato účelová funkce je tedy velmi citlivá na odchylky.

Funkce MSE je v podstatě hlavním optimalizačním algoritmem lineární regrese. Tedy metoda nejmenších čtverců. Ve výzkumné otázce byl tento problém řešen a došlo se k závěru, že lineární regrese není vhodná bez předchozího ověření statistických podmínek. Toto je v podstatě druhým důkazem, jelikož tento framework může vytvářet i lineární regrese. Použití účelové funkce MSE se tedy jeví jako nevhodné vzhledem k variabilitě jeho výsledků. Dále v práci bude použita hlavně účelová funkce LAD, která má zejména na datasetu D2 nižší variabilitu a dosahuje hlubších minim.

7.4 Výzkumná Otázka 4

V první fázi ověření této výzkumné otázky byl proveden výpočet počtu vzorků. K vyšetření této výzkumné otázky bude použit párový t-test. Nejprve tedy bylo nutné vytvořit pilotní vzorek, aby mohl být spočten počet vzorků, tak aby byla zachována statistická síla.

Tab. 7.9 Deskriptivní statistika pilotních 10 průchodů (MMRE)

	n	průměr	sd	medián	min	max	rozpětí	se
D1:Framework	10	37,80	18,77	30,26	18,85	73,54	54,69	5,94
D1:Karner	10	55,90	22,96	48,26	32,00	101,14	69,15	7,26
D2:Framework	10	8,54	1,04	1,24	6,75	9,98	-1,37	0,33
D2:Karner	10	27,50	2,71	2,47	21,87	30,29	-0,73	0,86

V tabulce 7.9 je vidět deskriptivní statistika pro pilotních 10 průchodů, aby jsme mohly spočítat kolik vzorků bude potřeba na dodržení síly testu. V tabulce je možné si povšimnout několika důležitých bodů. Pro dataset D1 se průměrná relativní chyba pro prezentovaný framework snížila o cca. 18 % oproti Karnerově rovnici. Avšak je zde vidět relativně velká směrodatná odchylka. Na datasetu D1 dosáhl framework minima 18,85 % MMRE, Karnerova rovnice 32 % MMRE. U datasetu D2 je situace velmi podobná, zde se průměr pro framework pohybuje na 8,5 % MMRE pro Karnerovu rovnici je to 27,5 % MMRE. Průměrné snížení relativní chyby je pro framework 19 %. Pro dataset D2 je směrodatná odchylka

nižší než pro dataset D1. Toto je způsobeno velikostí vzorků v obou datasetech. Minimum na datasetu D2 je pro framework 6,75 % MMRE pro Karnerovu rovnici je to 21,87 % MMRE.

Pilotní vzorky byly použity pro výpočet počtu vzorků pro statistické ověření. Pro dataset D1 je to 12 vzorků. Pro dataset D2 se jedná o 2 vzorky. Tak malé množství potřebných vzorků pro D2 je způsobeno velikostí datasetu D2, poměrně velkým rozdílem průměrů a zejména velmi nízkým rozptylem pilotního vzorku. Na směrodatné odchylce pro dataset D2 je vidět, že jsou rozdíly v průměru tak velké, že se výsledky nemohou k sobě přiblížit. Výpočet byl proveden v programu G*Power, jak je vidět na obrázku 7.7 a 7.8.

Výsledky tohoto experimentu jsou znázorněny na grafu 7.9. 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 chyby MMRE (bude statisticky dokázáno). 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.

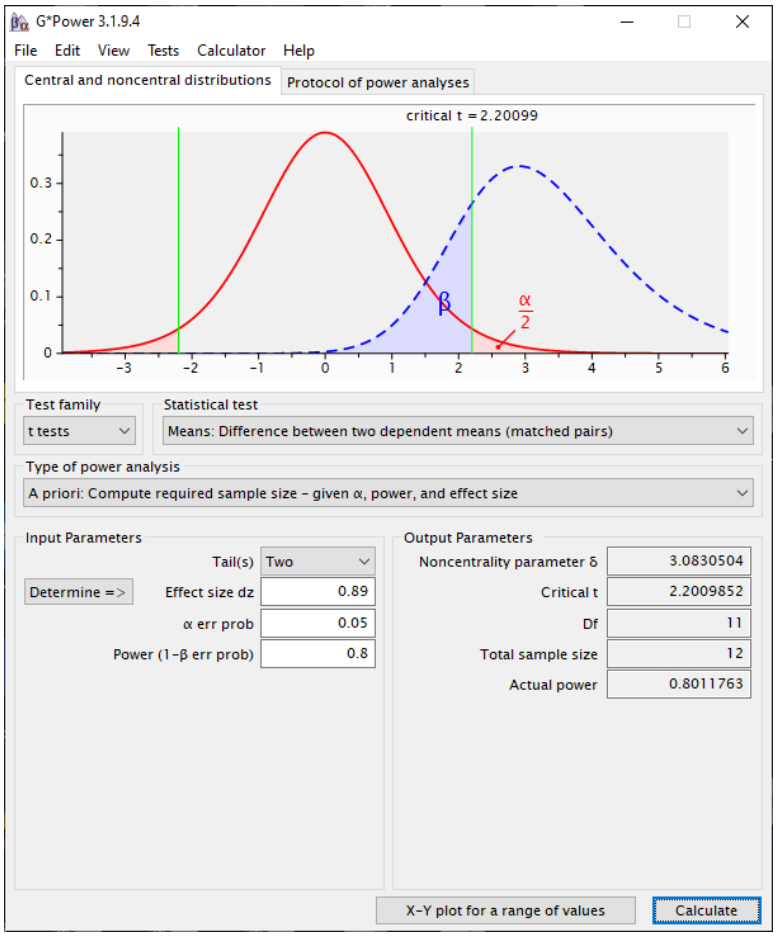
7.4.1 Statistické ověření

Pro ověření, zda prezentovaný framework statisticky významně zpřesňuje odhad bude použit jednostranný párový t-test. Bude použit párový t-test jelikož obě hodnoty MMRE byly vždy vypočteny pro stejná testovací data.

Tab. 7.10 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 7.10 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 statis-

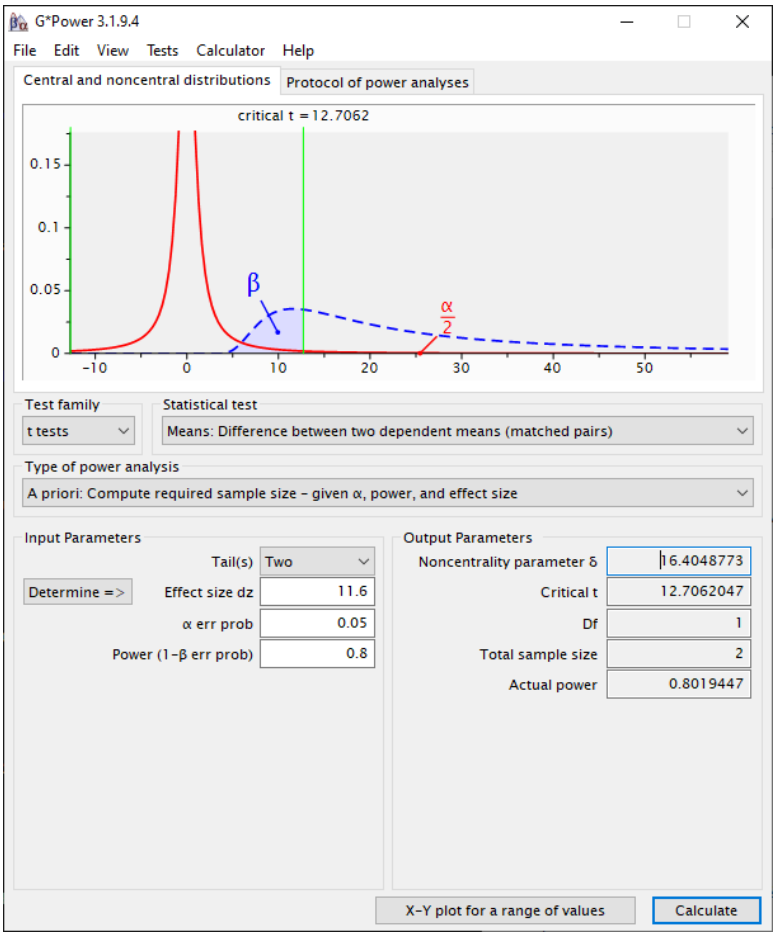


Obr. 7.7 Výpočet počtu vzorků pro výzkumnou otázku 4 datasetu D1, tak aby síla testu byla $1 - \beta > 80\%$

ticky významně nižší než pro standardní Karnerovu rovnici. Toto platí pro oba datasety tedy D1 i D2.

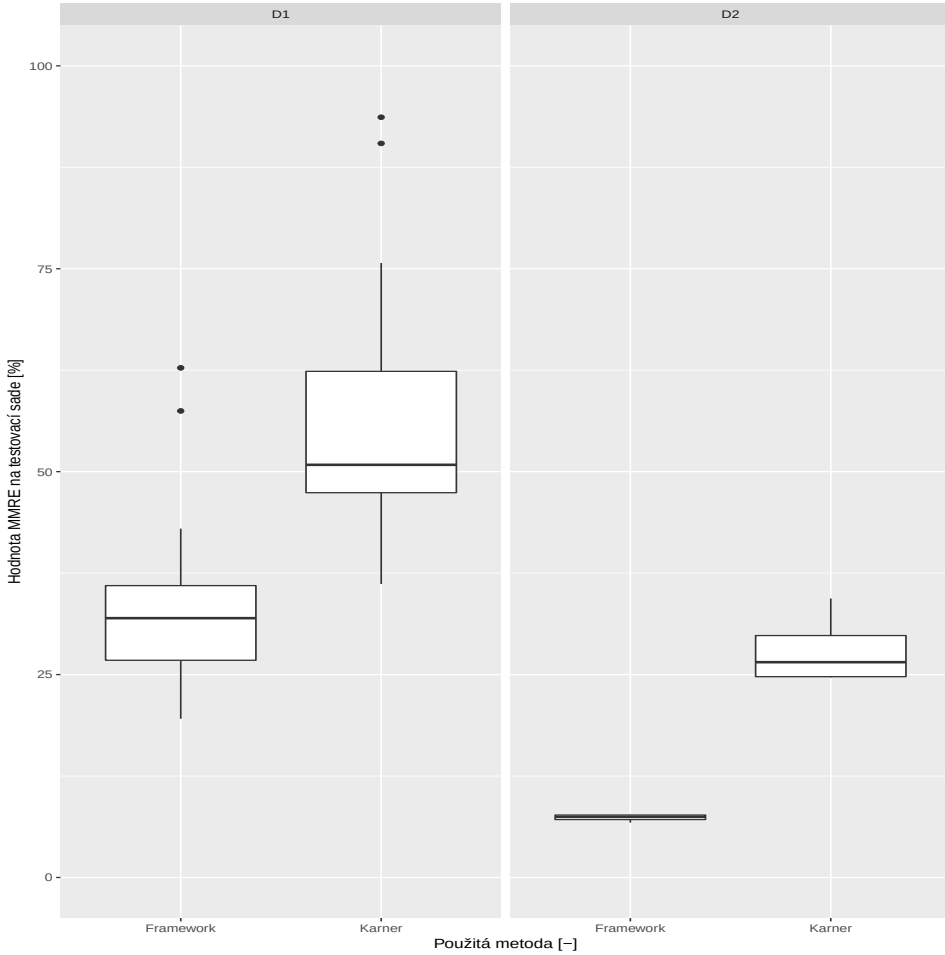
7.4.2 Shrnutí výsledků výzkumné otázky 4

Výsledky dosažené při vyšetřování této výzkumné otázky jsou pro prezentovaný 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



Obr. 7.8 Výpočet počtu vzorků pro výzkumnou otázku 4 datasetu D2, tak aby síla testu byla $1 - \beta > 80\%$

s faktorem produktivity 20 [47], což je poměrně standardní hodnota, zejména když se metoda UCP nastavuje pro novou softwarovou společnost. V současné době probíhá mnoho výzkumů, ve kterých jsou doporučovány jiné možné vhodnější avšak komplexnější hodnoty faktoru produktivity [90, 65, 104]. Můžeme tedy říci, že průměrné zpřesnění oproti Karnerově rovnici se v nejlepším případě pohybuje na vypočtených 18 % MMRE. Tedy za předpokladu, že faktor produktivity je standardně nastaven na hodnotu 20.

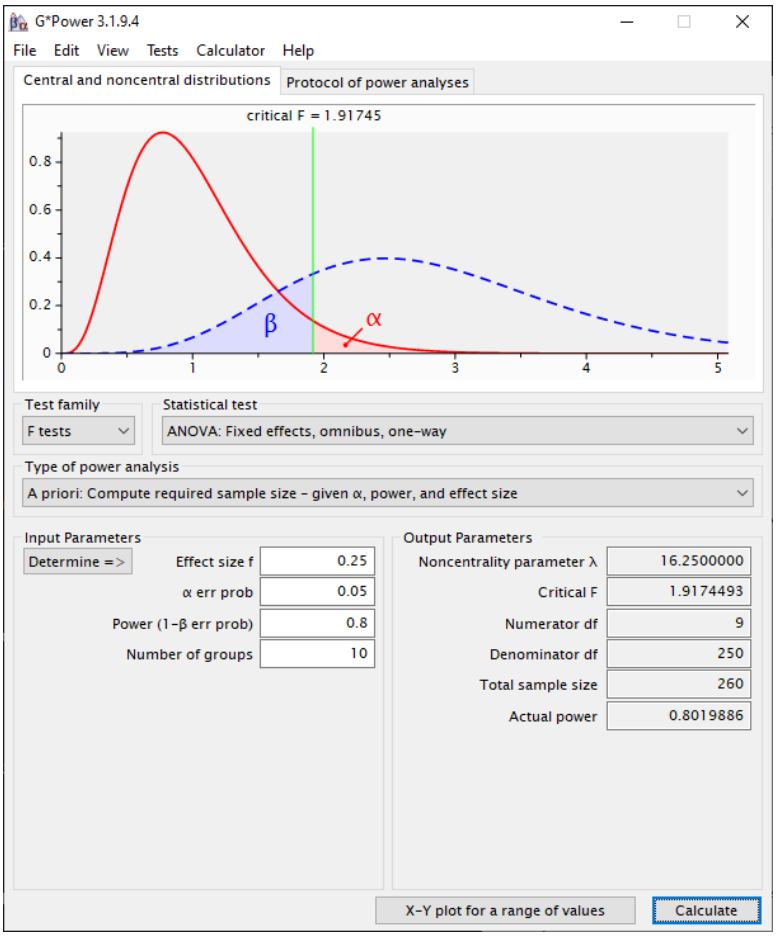


Obr. 7.9 Statistické porovnání prezentovaného frameworku a výpočtu pomocí Karnerovy rovnice

7.5 Výzkumná Otázka 5

V první fázi ověření této výzkumné otázky byl proveden výpočet počtu vzorků, tak, aby po použití testu ANOVA byla zachována statistická síla $1 - \beta > 0,8$. Po výpočtu je nutné získat dohromady 260 vzorků. Jelikož máme 10 skupin pro porovnání, je potřeba získat z každé skupiny (kombinace parametrů) 26 vzorků. Výpočet byl proveden v programu G*Power, jak je vidět na obrázku 7.10.

V tabulce 7.11 je vidět deskriptivní statistika pro použité modely na datasetu



Obr. 7.10 Výpočet počtu vzorků pro výzkumnou otázku 5, tak aby síla testu byla $1 - \beta > 80\%$

D1. Výsledky prezentovaného frameworku jsou na posledním řádku pod zkratkou frm. Pro každý model bylo nasbíráno 26 vzorků. Prezentovaný framework měl ze všech modelů nejnižší průměr a medián. Nejnižší minimum bylo zaznamenáno u krokové regrese (leapSeq). Maximum pak zaznamenáno u modelu rozhodovacího stromu CART (rpart). Grafické zobrazení získaných dat je vidět na obrázku 7.11.

Na grafu 7.11 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

Tab. 7.11 Deskriptivní statistika pro dataset D1
pro použité modely (MMRE)

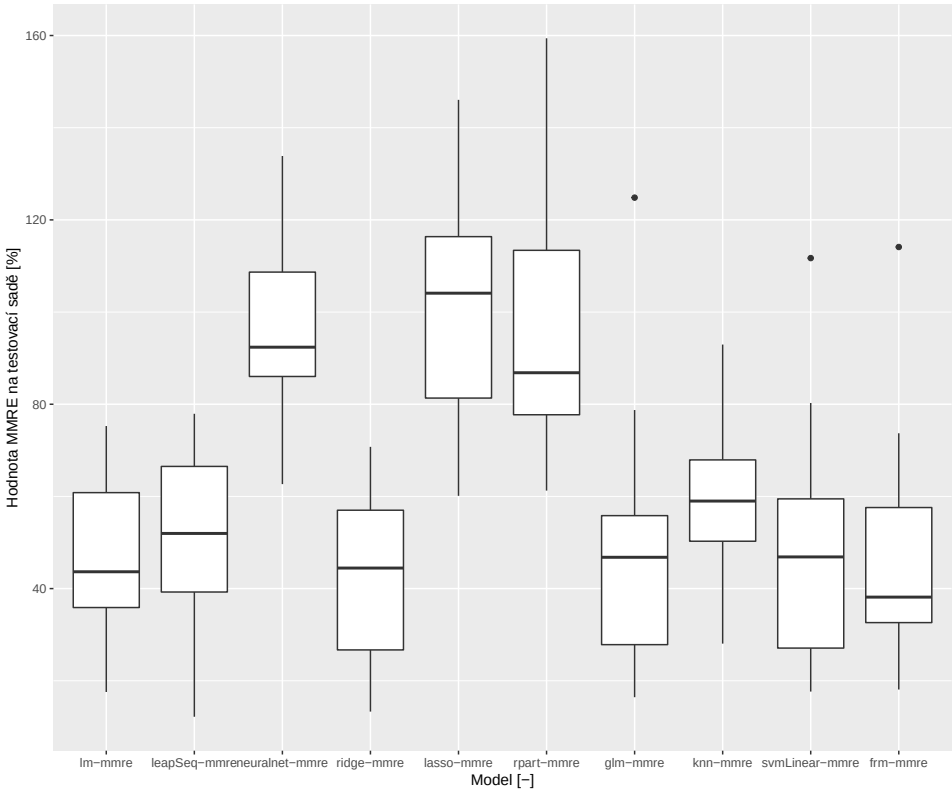
	n	průměr	sd	medián	min	max	rozpětí	se
lm	26	46,10	16,80	43,65	17,57	75,27	57,71	3,29
leapSeq	26	51,37	19,06	51,97	12,19	77,93	65,74	3,74
neuralnet	26	95,91	20,18	92,38	62,67	133,86	71,20	3,96
ridge	26	44,56	16,41	44,45	13,31	70,75	57,44	3,22
lasso	26	101,20	24,44	104,09	60,13	146,04	85,92	4,79
rpart	26	96,80	26,68	86,84	61,26	159,40	98,14	5,23
glm	26	45,17	23,05	46,79	16,43	124,83	108,40	4,52
knn	26	57,31	16,45	58,99	28,06	92,95	64,89	3,23
svm	26	46,06	22,51	46,87	17,65	111,71	94,06	4,41
frm	26	42,30	20,81	38,15	18,10	114,12	96,01	4,08

3 modelů a těmi jsou neuronová síť, LASSO regrese a rozhodovací strom. Nutno však podotknout, že zmíněné algoritmy nebyly nijak optimalizovány, byly použity standardní doporučované hodnoty.

Tab. 7.12 Deskriptivní statistika pro dataset D2
pro použité modely (MMRE)

	n	průměr	sd	medián	min	max	rozpětí	se
lm	26	6,63	1,29	6,37	4,88	9,39	4,51	0,25
leapSeq	26	6,45	0,76	6,41	4,42	8,08	3,66	0,15
neuralnet	26	8,95	0,38	8,91	8,38	9,81	1,42	0,07
ridge	26	6,72	1,00	6,97	4,85	8,77	3,93	0,20
lasso	26	8,17	0,41	8,17	7,42	8,73	1,31	0,08
rpart	26	8,96	0,32	9,00	8,40	9,61	1,21	0,06
glm	26	6,55	0,94	6,90	4,16	7,90	3,74	0,18
knn	26	4,41	0,84	4,24	2,69	6,13	3,44	0,16
svm	26	6,23	0,99	6,44	4,07	8,37	4,30	0,20
frm	26	8,08	0,83	7,76	6,64	9,98	3,34	0,16

V tabulce 7.12 je vidět deskriptivní statistika pro použité modely na datasetu D2. Výsledky prezentovaného frameworku jsou na posledním řádku pod zkratkou frm. Zde měl framework průměr 8 % MMRE. Nejlepší průměr dosahoval algoritmus k-nearest neighbors.



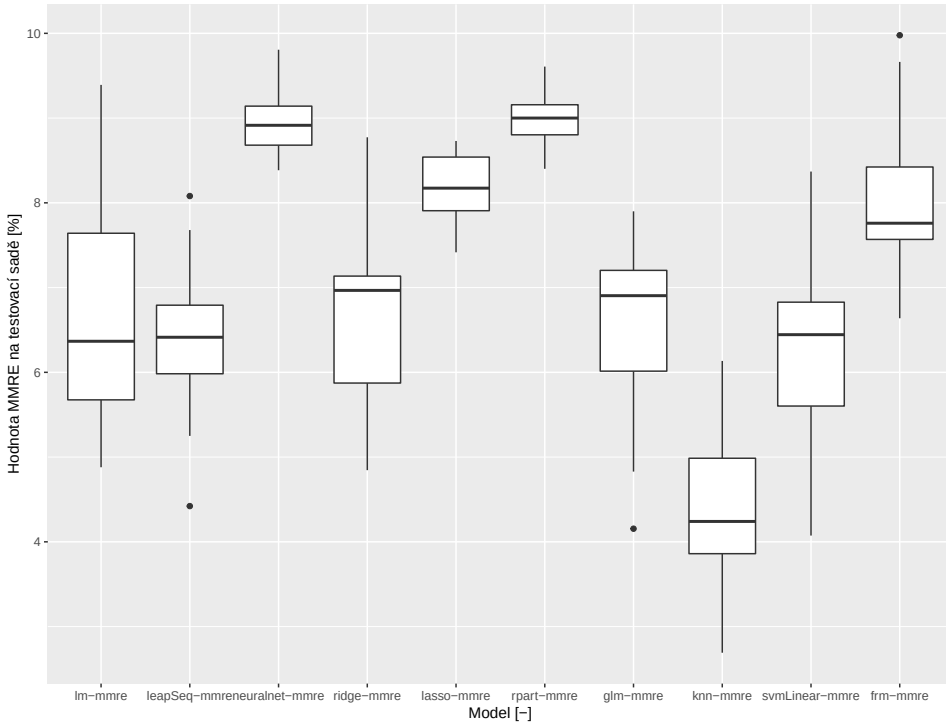
Obr. 7.11 Statistické porovnání prezentovaného frameworku a ostatních modelů na datasetu D1

Na grafu 7.12 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.

7.5.1 Statistické ověření

V tabulce 7.13 je vidět, že p-hodnota pro oba datasety je pod hodnotou 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.

V tabulce 7.14 je znázorněno zařazení do skupin Tukey post-hoc analýzy da-



Obr. 7.12 Statistické porovnání prezentovaného frameworku a ostatních modelů na datasetu D2

Tab. 7.13 Výpočet hypotéz ANOVA pro oba datasety

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

tasetu D1. Skupiny jsou vytvořeny na základě toho, zda vzájemné a vícenásobné porovnání dosažených výsledků jednotlivých metod pochází, na hladině významnosti 5 %, ze stejné populace. Zde je možné si povšimnout pouze dvou skupin "a" a "b". Skupina "a" zahrnuje modely neuronová síť, LASSO regrese a rozhodovací strom. Tyto modely, jak bylo ukázáno na grafu 7.11 měly horší statistické vlastnosti než ostatní modely. Ostatní modely včetně prezentovaného frameworku spadají do skupiny "b" a jsou tedy statisticky významně ekvivalentní.

Tab. 7.14 Post-hoc analýza ověření diferencí u datasetu D1

Parametr	Skupina
lm	b
leapSeq	b
neuralnet	a
ridge	b
lasso	a
rpart	a
glm	b
knn	b
svm	b
frm	b

Tab. 7.15 Post-hoc analýza ověření diferencí u datasetu D2

Parametr	Skupina
lm	c
leapSeq	c
neuralnet	a
ridge	c
lasso	b
rpart	a
glm	c
knn	d
svm	c
frm	b

V tabulce 7.15 je znázorněno zařazení do skupin Tukey post-hoc analýzy datasetu D2. Zde je možné si povšimnout pouze čtyř skupin "a","b","c" a "d". Skupina "a" zahrnuje modely neuronová síť a rozhodovací strom. Tyto modely, jak bylo ukázáno na grafu 7.11 měly nejhorší statistické vlastnosti. Do skupiny b byl zařazen prezentovaný framework a LASSO regrese. Regresní modely a "Support vector machine" spadají do kategorie "c". A algoritmus k-nearest neighbors jako jediný spadá do kategorie "d". Tento algoritmus dosahoval na daném data-

setu nejlepších výsledků.

7.5.2 Shrnutí výsledků výzkumné otázky 5

Výsledky dosažené při vyšetřování této výzkumné otázky jsou pro prezentovaný framework jedny z nejdůležitějších. 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.

8 Shrnutí dosažených výsledků

Všechny cíle disertační práce byly splněny. Následuje výčet dosažených výsledků pro každou výzkumnou otázku.

Existuje lineární závislost mezi ohodnocením softwarového projektu a skutečnou časovou náročností softwarového projektu?

Lineární modely dosahovaly na získaných datasetech velmi dobrých výsledků. Pokud se na dané modely díváme pouze z pohledu získání nejpřesnějších výsledků tzn. nezajímají nás statistické vlastnosti, tak je jejich použití možné. Avšak, jakmile začneme zkoumat jejich statistické vlastnosti zejména splnění Gauss-Markovových podmínek, potom cca. 30 % výběrů na datasetech nesplňovalo tyto podmínky a tedy použití lineární regrese ze statistického pohledu nebylo žádoucí. Pokud nejsou splněny Gauss-Markovovy podmínky, tak metoda nejmenších čtverců poskytuje nekonzistentní odhady [106]. Uvažovat apriori o linearitě, bez ověření jejich podmínek není možné.

Které parametry metody UCP mají největší vliv na přesnější odhad při použití nového frameworku?

Jak bylo ukázáno ve výzkumné otázce 2 nejdůležitějším parametrem je UUCW. Toto se ukázalo, ale jen na menším datasetu D1. Důvod, proč toto nebylo ukázáno na datasetu D2 je zřejmě ten, že vyžadování parametru nebylo podmíněno. Bylo pouze zakázáno použití ostatních parametrů. Mohlo se tedy stávat a jistě se stávalo u datasetu D2, že byla vygenerována pouze konstanta bez parametru.

Jaký je vliv různých účelových funkcí na přesnost odhadu a která účelová funkce poskytne nejpřesnější odhad?

Ve výzkumné otázce 3, bylo ukázáno, že nejvíce vhodnou je účelová funkce LAD. Na menším datasetu sice nejnižších minim dosahovala funkce MSE, nicméně na datasetu D2 si funkce MSE vedla podstatně hůře než ostatní použité funkce. Důvody by mohlo být, že funkce MSE pracuje s mocninou chyby. Vzhledem k této matematické vlastnosti přikládá vzdálenějším pozorováním větší váhu. Tato funkce je tedy citlivější na odchylky od očekávané hodnoty. Z tohoto důvodu byla ve zbytku práce preferována účelová funkce LAD, která počítá s absolutní hodnotou, a tedy všechny pozorovaní pro ni mají stejnou váhu.

Je odhad pomocí nového frameworku přesnější než odhad podle standardní UCP rovnice?

Ano, v řešení výzkumné otázky 4 bylo ukázáno zpřesnění na obou datasetech o cca. 18 % ve prospěch nového frameworku.

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

Ano, ale pouze u některých modelů. Jmenovitě jsou to neuronová síť, rozhodovací strom a LASSO regrese. Na menším datasetu si prezentovaný framework však vedl v průměru lépe než ostatní modely, a dokonce i medián byl nejnižší. Nicméně statisticky se nepodařilo dokázat, že je toto zpřesnění signifikantní. I tak se nový framework zařadil mezi nejlepší modely pro danou úlohu. Na datasetu D2 si framework vedl hůře. Největším faktorem jsou použité matematické funkce. Některé z uvedených modelů totiž nejsou jen matematické funkce, ale mají možnost rozhodování. Matematické funkce byly použity z důvodů jejich interpretovatelnosti a větší přesnosti na spojitých datech. Dataset D2 je větší a možnosti generování funkcí se snižují v limitním případě až na konstantu. Zde mají výhodu neparametrické modely jako k-nearest neighbors. Nicméně, pro dataset D1, model k-nearest neighbors byl zaznamenán jako čtvrtý nejméně přesný model.

9 Další možný rozvoj a otevřené otázky

Pro návrh frameworku byla použita metoda UCP, avšak framework je navržen modulárním způsobem, a není proto vyloučeno místo metody odhadování základních charakteristik projektů UCP použít jinou odhadovací metodu pro příklad COCOMO či FP.

Otevřenou otázkou je "jaký je nejmenší počet projektů, aby mohla být tato metoda použita?". Je to metoda řízená daty, a tak je potřeba mít k dispozici datovou sadu historických projektů. Samozřejmě jsou k dispozici doporučení jak velký by měl být poměr počtu projektů v závislosti na počtu nezávislých proměnných. Pro framework DE a AP jsou tyto doporučení minimálně diskutabilní. Důvod je ten, že u většiny modelů strojového učení je jejich funkční struktura předem známá. Pro framework DE a AP toto neplatí. Ve výzkumné otázce 5 je uvedena komparativní studie jejíž závěry jsou, že na menším datasetu D1 je framework v průměru relativní chyby přesnější než ostatní použité modely.

Další možným rozvojem navrhovaného odhadovacího frameworku je použití algoritmu SHADE. Nevýhodou použitého frameworku je poměrně náročné nastavení parametrů jak diferenciální evoluce, tak analytického programování. Při nesprávném nastavení může docházet zejména k velké výpočetní náročnosti, čemuž by mohl algoritmus SHADE zabránit.

Další otevřenou otázkou je určení funkcí (GFS) pro algoritmus analytického programování. V této práci byly použity standardní matematické funkce. Nicméně algoritmus analytického programování je schopen pracovat také s funkcí IF a jiných programátorsky definovaných konstruktů. Autor zvolil standardní matematické funkce, hlavně z důvodu jejich interpretovatelnosti, avšak nepopírá, že jiné funkce by dokázaly odhad ještě více zpřesnit.

10 Přínos práce

10.1 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 [70]. 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.

10.2 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.

11 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 %.

SEZNAM POUŽITÉ LITERATURY

[1] Abdelali, Z., Mustapha, H. and Abdelwahed, N. Investigating the use of random forest in software effort estimation. In *Procedia Computer Science*, 148, 2019. doi: 10.1016/j.procs.2019.01.042.

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

[3] Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician*. 1992, 46, 3. ISSN 15372731. doi: 10.1080/00031305.1992.10475879.

[4] Anastasiadis, A. D., Magoulas, G. D. and Vrahatis, M. N. New globally convergent training scheme based on the resilient propagation algorithm. *Neurocomputing*. 2005, 64, 1-4 SPEC. ISS. ISSN 09252312. doi: 10.1016/j.neucom.2004.11.016.

[5] Barrett, B. E. and Mellichamp, J. M. Software Development Cost Estimation Using Function Points. *IEEE Transactions on Software Engineering*. 1994. ISSN 00985589. doi: 10.1109/32.277575.

[6] 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.

[7] Bermejo, S. and Cabestany, J. Oriented principal component analysis for large margin classifiers. *Neural Networks*. 2001, 14, 10, pp. 1447–1461. ISSN 08936080. doi: 10.1016/S0893-6080(01)00106-X.

[8] 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. doi: 10.1109/CEC.2001.934438.

[9] Boehm, W. Software Engineering Economics. *IEEE Transactions on Software Engineering*. jan 1984, SE-10, 1, pp. 4–21. ISSN 0098-5589. doi: 10.1109/TSE.1984.5010193.

[10] Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. *Classification and regression trees*. 2017. doi: 10.1201/9781315139470.

[11] Brereton, R. G. and Lloyd, G. R. Support Vector Machines for classification and regression, 2010. ISSN 13645528.

[12] 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. doi: 10.1109/ICSE.1998.671392. ISBN 0-8186-8368-6.

[13] 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). doi: 10.1016/S0950-5849(01)00192-6.

[14] Cerpa, N., Bardeen, M., Astudillo, C. A. and Verner, J. Evaluating different families of prediction methods for estimating software project outcomes. In *Journal of Systems and Software*, 2016. doi: 10.1016/j.jss.2015.10.011.

[15] Chambers J. M. Linear Models. California: Wadsworth & Brooks/Cole, 1992.

[16] Chang, C. C. and Lin, C. J. LIBSVM: A Library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2011, 2, 3. ISSN 21576904. doi: 10.1145/1961189.1961199.

[17] Chavoya, A., Lopez-Martin, C. and Meda-Campaña, M. E. Software Development Effort Estimation by Means of Genetic Programming. *International Journal of Advanced Computer Science and Applications(IJACSA)*. 2013, 4, 11.

[18] Chen, Y. Y., Lin, Y. H., Kung, C. C., Chung, M. H. and Yen, I. H. Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes. *Sensors (Switzerland)*. 2019, 19, 9. ISSN 14248220. doi: 10.3390/s19092047.

[19] Citac and Eurachem. Quantifying Uncertainty in Analytical Measurement. *English*. 2000. ISSN 0-94948926-12-0. doi: 0948926155.

[20] Collector, D. and Module, F. G. Qualitative Research Methods Overview. *Qualitative Research Methods A Data Collectors Field Guide*. 2011. ISSN 00222437. doi: 10.2307/3172595.

[21] Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F. and Mendes, E. Using tabu search to configure support vector regression for effort estimation. *Empirical Software Engineering*. 2013, 18, 3. ISSN 13823256. doi: 10.1007/s10664-011-9187-3.

[22] Dejaeger, K., Verbeke, W., Martens, D. and Baesens, B. Data mining techniques for software effort estimation: A comparative study. *IEEE Transactions on Software Engineering*. 2012. ISSN 00985589. doi: 10.1109/TSE.2011.55.

[23] Demirörs, O. and Gencel, Ç. A comparison of size estimation techniques applied early in the life cycle. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2004. ISSN 03029743.

[24] Dobson, A. J. and Barnett, A. G. *An introduction to generalized linear models, third edition*. 2008. doi: 10.1080/02664760802695900.

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

[26] Dreyfus, S. E. Artificial neural networks, back propagation, and the kelley-bryson gradient procedure. *Journal of Guidance, Control, and Dynamics*. 1990, 13, 5. ISSN 07315090. doi: 10.2514/3.25422.

[27] Fan, R. E., Chen, P. H. and Lin, C. J. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*. 2005, 6. ISSN 15337928.

[28] Faul, F., Erdfelder, E., Lang, A. G. and Buchner, A. G*Power 3: A flexible statistical power analysis program for the social, behavioral,

and biomedical sciences. In *Behavior Research Methods*, 39, pp. 175–191. Psychonomic Society Inc., 2007. doi: 10.3758/BF03193146.

[29] Fedotova, O., Teixeira, L., Alvelos and Helena. Software effort estimation with multiple linear regression: Review and practical application. *Journal of Information Science and Engineering*. 2013, 29, 5. ISSN 10162364. doi: 10.6688/JISE.2013.29.5.8.

[30] Foss, T., Stensrud, E., Kitchenham, B. and Myrtveit, I. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering*. 2003. ISSN 00985589. doi: 10.1109/TSE.2003.1245300.

[31] Freedman, D. A. *Statistical models: Theory and practice*. 2009. doi: 10.1017/CBO9780511815867.

[32] Golafshani, N. Understanding reliability and validity in qualitative research. *The Qualitative Report*. 2003. ISSN 14754762. doi: 10.3367/UFNr.0180.201012c.1305.

[33] Gray, R. J. Flexible Methods for Analyzing Survival Data Using Splines, With Applications to Breast Cancer Prognosis. *Journal of the American Statistical Association*. dec 1992, 87, 420, pp. 942. ISSN 01621459. doi: 10.2307/2290630.

[34] Grenning, J. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*. 2002, , April, pp. 1–3.

[35] Haldar, S. and Miller, A. J. Subset Selection in Regression. *Journal of Marketing Research*. 1992, 29, 2. ISSN 00222437. doi: 10.2307/3172576.

[36] Hansheng, W., Guodong, L. and Tsai, C. L. Regression coefficient and autoregressive order shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*. 2007, 69, 1. ISSN 14679868. doi: 10.1111/j.1467-9868.2007.00577.x.

[37] Hastie, T., Tibshirani, R. and Friedman, J. The Elements of Statistical Learning. *Bayesian Forecasting and Dynamic Models*. 2009, 1, pp. 1–694. ISSN 0172-7397. doi: 10.1007/b94608.

[38] Heemstra, F. J. Software cost estimation. *Information and Software Technology*. 1992. ISSN 09505849. doi: 10.1016/0950-5849(92)90068-Z.

[39] Hocking, R. R. The analysis and selection of variables in linear regression. *Biometrics*. 1976, 32, 1. ISSN 0006-341X. doi: 10.2307/2529336.

[40] Hoerl, A. E. and Kennard, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*. 1970, 12, 1. ISSN 15372723. doi: 10.1080/00401706.1970.10488634.

[41] Huang, G. B., Zhu, Q. Y. and Siew, C. K. Extreme learning machine: Theory and applications. *Neurocomputing*. 2006, 70, 1-3. ISSN 09252312. doi: 10.1016/j.neucom.2005.12.126.

[42] Hughes, R. T. Expert judgement as an estimating method. *Information and Software Technology*. jan 1996, 38, 2, pp. 67–75. ISSN 09505849. doi: 10.1016/0950-5849(95)01045-9.

[43] Idri, A., Abnane, I. and Abran, A. Support vector regression-based imputation in analogy-based software development effort estimation. *Journal of Software: Evolution and Process*. 2018, 30, 12. ISSN 20477481. doi: 10.1002/smr.2114.

[44] James, G., Witten, D., Hastie, T. and Tibishirani, R. *An Introduction to Statistical Learning with Applications in R (older version)*. 2013. doi: 10.1007/978-1-4614-7138-7.

[45] Jenkins, A. M., Naumann, J. D. and Wetherbe, J. C. Empirical investigation of systems development practices and results. *Information and Management*. 1984. ISSN 03787206. doi: 10.1016/0378-7206(84)90012-0.

[46] Jorgensen, M. and Shepperd, M. A systematic review of software development cost estimation studies, 2007. ISSN 00985589.

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

[48] 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. doi: 10.1109/APSEC.2008.43.

[49] Kitchenham, B. a., MacDonell, S. G., Pickard, L. and Shepperd, M. J. What accuracy statistics really measure. *IEE Proceedings – Software Engineering*. 2001, 148, pp. 81–85. ISSN 14625970. doi: 10.1049/ip-sen:20010506.

[50] Kneuper, R. and Kneuper, R. Selected Current Trends in Software Processes. In *Software Processes and Life Cycle Models*. 2018.

[51] Knuth, D. E. backus normal form vs. Backus Naur form. *Communications of the ACM*. 1964. ISSN 15577317. doi: 10.1145/355588.365140.

[52] 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. doi: 10.1016/j.jss.2013.02.053.

[53] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference of Artificial Intelligence*. 1995.

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

[55] Koza, J. Tutorial on advanced genetic programming, at genetic programming. 1997.

[56] Krishnan, M. Against Interpretability: a Critical Examination of the Interpretability Problem in Machine Learning. *Philosophy and Technology*. 2019. ISSN 22105441. doi: 10.1007/s13347-019-00372-9.

[57] Lavazza, L. and Morasca, S. Software effort estimation with a generalized robust linear regression technique. In *IET Seminar Digest*, 2012, 2012. doi: 10.1049/ic.2012.0027.

[58] 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.

[59] Mahboob, T., Gull, S., Ehsan, S. and Sikandar, B. Predictive Approach towards Software Effort Estimation using Evolutionary Support Vector Machine. *International Journal of Advanced Computer Science and Applications*. 2017, 8, 5. ISSN 2158107X. doi: 10.14569/ijacsa.2017.080554.

[60] Myrtveit, T. F., Stensrud, E., Kitchenham, B. and Ingunn. A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering*. 2003, 29, pp. 1–30. ISSN 0098-5589. doi: 10.1109/TSE.2003.1245300.

[61] Najm, A., Zakrani, A. and Marzak, A. Decision Trees Based Software Development Effort Estimation: A Systematic Mapping Study. In *Proceedings of 2019 International Conference of Computer Science and Renewable Energies, ICCSRE 2019*, 2019. doi: 10.1109/ICCSRE.2019.8807544.

[62] Nassif, a. B., Capretz, L. F. and Ho, D. Estimating Software Effort Based on Use Case Point Model Using Sugeno Fuzzy Inference System. *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*. 2011, pp. 393–398. ISSN 1082-3409. doi: 10.1109/ICTAI.2011.64.

[63] Nassif, A. B., Ho, D. and Capretz, L. F. Regression model for software effort estimation based on the use case point method. *2011 International Conference on Computer and Software Modeling*. 2011, 14, January, pp. 117–121.

[64] Nassif, A. B., Capretz, L. F. and Ho, D. Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade

Correlation Neural Network Model. In *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 589–594. IEEE, aug 2012. doi: 10.1109/SNPD.2012.40. ISBN 978-1-4673-2120-4.

[65] Nassif, A. B., Ho, D. and Capretz, L. F. Towards an early software estimation using log-linear regression and a multilayer perceptron model. *Journal of Systems and Software*. jan 2013, 86, 1, pp. 144–160. ISSN 01641212. doi: 10.1016/j.jss.2012.07.050.

[66] Nguyen, V., Steece, B. and Boehm, B. A constrained regression technique for COCOMO calibration. In *ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2008. doi: 10.1145/1414004.1414040.

[67] 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. doi: 10.1016/j.infsof.2010.10.005.

[68] Oliveira, A. L. Estimation of software project effort with support vector regression. *Neurocomputing*. 2006, 69, 13-15. ISSN 09252312. doi: 10.1016/j.neucom.2005.12.119.

[69] Olu-Ajayi, R. An Investigation into the Suitability of k-Nearest Neighbour (k-NN) for Software Effort Estimation. *International Journal of Advanced Computer Science and Applications*. 2017, 8, 6. ISSN 2158107X. doi: 10.14569/ijacsa.2017.080628.

[70] 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.

[71] Oplatkova, Z. K., Senkerik, R., Zelinka, I. and Pluhacek, M. Analytic programming in the task of evolutionary synthesis of a controller for

high order oscillations stabilization of discrete chaotic systems. *Computers & Mathematics with Applications*. aug 2013, 66, 2, pp. 177–189. ISSN 0898-1221.

[72] Pai, D. R., McFall, K. S. and Subramanian, G. H. Software effort estimation using a neural network ensemble. *Journal of Computer Information Systems*. 2013, 53, 4. ISSN 08874417. doi: 10.1080/08874417.2013.11645650.

[73] Papatheocharous, E., Papadopoulos, H. and Andreou, A. S. Software Effort Estimation with Ridge Regression and Evolutionary Attribute Selection. *arXiv preprint arXiv\;1012.5754*. 2010.

[74] Patton, M. Qualitative Evaluation and Research Methods. *Qualitative Evaluation and Research Methods*. 1990. ISSN 01606891. doi: 10.1002/nur.4770140111.

[75] Pengelly, A. Performance of effort estimating techniques in current development environments. *Software engineering journal*. 1995. ISSN 02686961. doi: 10.1049/sej.1995.0022.

[76] Pliskin, J. and Kennedy, P. A Guide to Econometrics. *Journal of the American Statistical Association*. 1987, 82, 399. ISSN 01621459. doi: 10.2307/2288828.

[77] Port, D. and Korte, M. Comparative studies of the model evaluation criterions MMRE and PRED in software cost estimation research. In *ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 51–60, 2008. doi: 10.1145/1414004.1414015. ISBN 9781595939715.

[78] Pramod Kumar, M., Babu Reddy, M. and Lakshmi Lavanya, A. An empirical approach to predict software development effort using linear regression model. *ARPN Journal of Engineering and Applied Sciences*. 2018, 13, 7. ISSN 18196608.

[79] Qi, F., Jing, X. Y., Zhu, X., Xie, X., Xu, B. and Ying, S. Software effort estimation based on open source projects: Case study of Github.

Information and Software Technology. 2017, 92. ISSN 09505849. doi: 10.1016/j.infsof.2017.07.015.

[80] Quinlan, J. R. Induction of Decision Trees. *Machine Learning*. 1986, 1, 1. ISSN 15730565. doi: 10.1023/A:1022643204877.

[81] Rao, B., Sameet, B. and Swathi, G. A novel neural network approach for software cost estimation using Functional Link Artificial Neural Network (FLANN). ... *Journal of Computer* .... 2009, 9, 6.

[82] Rasool, R. and Malik, A. A. Effort estimation of ETL projects using Forward Stepwise Regression. In *Proceedings of 2015 International Conference on Emerging Technologies, ICET 2015*, 2016. doi: 10.1109/ICET.2015.7389209.

[83] Riedmiller, M. Rprop-description and implementation details. Technical report, 1994.

[84] Riedmiller, M. and Braun, H. Direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *1993 IEEE International Conference on Neural Networks*, 1993. doi: 10.1109/icnn.1993.298623.

[85] Ripley, B. D. *Pattern recognition and neural networks*. 2014. doi: 10.1017/CBO9780511812651.

[86] Ryan, C., Collins, J. J. and Neill, M. O. Grammatical evolution: Evolving programs for an arbitrary language. In Banzhaf, W., Poli, R., Schoenauer, M. and Fogarty, T. C. (Ed.) *Genetic Programming: First European Workshop, EuroGP'98 Paris, France, April 14–15, 1998 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. pp. 83–96. doi: 10.1007/BFb0055930. ISBN 978-3-540-69758-9.

[87] Salzberg, S. L. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*. 1997, 1, 3, pp. 317–328. ISSN 13845810. doi: 10.1023/A:1009752403260.

[88] Satapathy, S. M. and Rath, S. K. Empirical assessment of machine learning models for effort estimation of web-based applications. In *ACM*

International Conference Proceeding Series, 2017. doi: 10.1145/3021460.3021468.

[89] Satapathy, S. M., Kumar, M. and Rath, S. K. Class point approach for software effort estimation using soft computing techniques. In *Proceedings of the 2013 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2013*, 2013. doi: 10.1109/ICACCI.2013.6637167. ISBN 9781467362153.

[90] Schneider, G. and Winters, J. *Applying Use Cases: A Practical Guide*. Addison-Wesley Professional, 2001. ISBN 978-0201708530.

[91] 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. doi: 10.1109/ICCCAS.2002.1178979.

[92] Shepperd, M., Cartwright, M. and Kadoda, G. On building prediction systems for software engineers. *Empirical Software Engineering*. 2000, 5, pp. 175–182. ISSN 13823256. doi: 10.1023/A:1026582314146.

[93] 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.

[94] Silhavy, P., Silhavy, R. and Prokopova, Z. Categorical variable segmentation model for software development effort estimation. *IEEE Access*. 2019, 7. ISSN 21693536. doi: 10.1109/ACCESS.2019.2891878.

[95] 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.

[96] 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. doi: 10.1371/journal.pone.0141887.

[97] Singh, Y., Kaur, A., Bhatia, P. K. and Sangwan, O. Predicting software development effort using artificial neural network. *International Journal of Software Engineering and Knowledge Engineering*. 2010, 20, 3. ISSN 02181940. doi: 10.1142/S0218194010004761.

[98] Snell, J., Birkes, D. and Dodge, Y. Alternative Methods of Regression. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*. 1996, 159, 1, pp. 182. ISSN 09641998. doi: 10.2307/2983483.

[99] Stensrud, E., Foss, T., Kitchenham, B. and Myrtveit, I. A further empirical investigation of the relationship between MRE and project size. *Empirical Software Engineering*. 2003, 8, 2, pp. 139–161. ISSN 13823256. doi: 10.1023/A:1023010612345.

[100] Štolfa, J., Koberský, O., Kopka, M., Krömer, P., Štolfa, S., Kožusznik, J. and Snášel, V. Value estimation of the use case parameters using SOM and fuzzy rules. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES 2012*, 2012. doi: 10.1145/2457276.2457305. ISBN 9781450317559.

[101] Storn, R. and Price, K. *Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces*. 11. Technical Report TR-95-012, 1995. doi: 10.1023/A:1008202821328. ISBN TR-95-012.

[102] 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. doi: 10.1023/A:1008202821328.

[103] Štrba, R., Štolfa, S., Štolfa, J., Vondrák, I. and Snášel, V. An application of neural network in method for use case based effort estimation. In *Frontiers in Artificial Intelligence and Applications*, 2017. doi: 10.3233/978-1-61499-720-7-231. ISBN 9781614997191.

[104] 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.

[105] The Standish Group International. CHAOS Report 2015. Technical report, 2015.

[106] Theil, H. *Principles of Econometrics*. New York : Wiley, 1971. ISBN 0-471-85845-5.

[107] Tibshirani, R. Regression shrinkage and selection via the lasso: A retrospective. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*. 2011, 73, 3. ISSN 13697412. doi: 10.1111/j.1467-9868.2011.00771.x.

[108] Tran, T., Nguyen, V., Truong, T., Tran, C. and Le, P. An evaluation of parameter pruning approaches for software estimation. In *ACM International Conference Proceeding Series*, 2019. doi: 10.1145/3345629.3345633.

[109] Trendowicz, A. and Jeffery, R. *Software Project Effort Estimation*. 2014. doi: 10.1007/978-3-319-03629-8. ISBN 9783319036281.

[110] Tsakonas, A. and Dounias, G. Deriving Models for Software Project Effort Estimation by Means of Genetic Programming. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, {KDIR} 2009*, pp. 34–42, 2009.

[111] Tung Khuat and Hanh Le. An Effort Estimation Approach for Agile Software Development using Fireworks Algorithm Optimized Neural Network. *International Journal of Computer Science and Information Security*. 2016, 14, 7.

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

[113] Van Solingen, R. Measuring the ROI of software process improvement. *IEEE Software*. 2004. ISSN 07407459. doi: 10.1109/MS.2004.1293070.

[114] Venables, W. N. and Ripley, B. D. Modern Applied Statistics with S-Plus. *Biometrics*. 1996, 52, 4. ISSN 0006341X. doi: 10.2307/2532871.

[115] Venables, W. N. and Ripley, B. D. Statistics Complements to Modern Applied Statistics with S. In *Modern Applied Statistics with S*. 2002.

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

[117] Wilkinson, G. N. and Rogers, C. E. SYMBOLIC DESCRIPTION OF FACTORIAL MODELS FOR ANALYSIS OF VARIANCE. *Journal of Applied Statistics*. 1973, 22, 3. ISSN 02664763. doi: 10.2307/2346786.

[118] Wu, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*. 2008. ISSN 02191377. doi: 10.1007/s10115-007-0114-2.

[119] Xu, Z. and Khoshgoftaar, T. M. Identification of fuzzy models of software cost estimation. *Fuzzy Sets and Systems*. 2004. ISSN 01650114. doi: 10.1016/j.fss.2003.10.008.

[120] Yurdakurban, V. and Erdoğan, N. Comparison of machine learning methods for software project effort estimation. In *26th IEEE Signal Processing and Communications Applications Conference, SIU 2018*, 2018. doi: 10.1109/SIU.2018.8404495.

[121] Zakrani, A., Hain, M. and Idri, A. Improving software development effort estimation using support vector regression and feature selection. *IAES International Journal of Artificial Intelligence*. 2019, 8, 4. ISSN 22528938. doi: 10.11591/ijai.v8.i4.pp399-410.

[122] 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.

[123] 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.

[124] Ziegel, E. R. An Introduction to Generalized Linear Models. *Technometrics*. 2002, 44, 4. ISSN 0040-1706. doi: 10.1198/tech.2002.s91.

[125] Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*. 2005, 67, 2. ISSN 13697412. doi: 10.1111/j.1467-9868.2005.00503.x.

SEZNAM PUBLIKACÍ AUTORA

[P.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.

[P.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.

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

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

[P.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.

[P.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.

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

[P.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.

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

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

[P.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.

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

[P.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.

[P.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.

[P.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.

[P.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.

[P.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.

ŽIVOTOPIS

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 Univerzita Tomáše Bati, 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

Komunikační dovednosti
Dobré komunikační dovednosti získané během výuky předmětů na doktorském studiu

Organizační/manažerské dovednosti
Vedení a organizace výuky

Digitální dovednosti
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

Další dovednosti

Řidičský průkaz B
Specializovaný statistický software : R, Stan, JAGS, WebPPL, Church

DOPLŇUJÍCÍ INFORMACE

H-index 3

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

Vědecké zájmy :
Softwarové inženýrství
Umělá inteligence
Optimalizace
Statistika
Pravděpodobnostní programování
Teorie rozhodování

SEZNAM PŘÍLOH

PŘÍLOHA A: DATASET D1

ID	UUCW	UAW	TCF	ECF	Effort [čh]
1	195	12	0.78	0.78	3037
2	80	10	0.75	0.81	1917
3	75	6	0.90	1.05	1173
4	130	9	0.85	0.89	742
5	85	12	0.82	0.79	614
6	50	9	0.85	0.88	492
7	50	6	0.78	0.51	277
8	305	14	0.94	1.02	3593
9	85	12	1.03	0.80	1681
10	130	12	0.71	0.73	1344
11	80	9	1.05	0.95	1220
12	70	12	0.78	0.79	720
13	30	4	0.96	0.96	514
14	100	15	0.90	0.91	397
15	355	15	1.12	0.77	3684
16	145	18	1.08	0.77	1980
17	325	12	1.09	0.94	3950
18	90	6	1.08	1.08	1925
19	125	9	1.02	0.98	2175
20	120	9	1.11	0.99	2226
21	200	12	1.00	0.92	2640
22	175	9	0.95	0.92	2568
23	245	12	0.89	1.19	3042
24	140	6	0.96	0.76	1696

PŘÍLOHA B: DATASET D2

Id	U	A	T	E	Effort
1	355	9	0.81	0.84	7970
2	445	8	0.99	0.99	7962
3	355	9	1.03	0.80	7935
4	350	8	0.90	0.91	7805
5	345	8	0.90	0.91	7758
6	345	8	0.99	0.99	7643
7	420	7	0.94	1.02	7532
8	340	8	1.03	0.80	7451
9	415	7	1.02	0.98	7427
10	335	8	0.92	0.78	7406
11	330	8	0.85	0.89	7365
12	410	7	0.75	0.81	7350
13	405	7	1.02	1.08	7303
14	325	8	1.09	0.95	7252
15	325	8	0.92	0.78	7245
16	400	7	0.75	0.81	7166
17	390	7	0.96	0.76	7119
18	315	8	0.92	0.78	7111
19	315	8	1.05	0.95	7044
20	390	7	0.71	0.73	7040
21	395	7	1.02	1.08	7028
22	315	8	1.03	0.80	6942
23	305	7	0.96	0.76	6814
24	380	9	0.78	0.79	6809
25	375	8	0.98	0.97	6802
26	305	12	0.78	0.51	6787
27	305	7	1.08	0.77	6764
28	380	7	1.05	0.95	6761
29	300	7	0.85	0.89	6725
30	300	7	1.02	1.08	6690
31	300	7	1.08	0.77	6600
32	290	7	0.94	1.02	6474
33	355	6	0.95	0.92	6433
34	285	7	0.78	0.79	6416
35	290	8	0.94	1.02	6412
36	355	6	0.90	0.94	6400
37	285	7	0.71	0.73	6360
38	355	6	0.90	0.91	6337
39	610	18	0.72	0.67	6240
40	280	7	0.78	0.51	6232
41	280	7	1.03	0.80	6173
42	345	6	1.00	0.92	6160
43	340	6	1.09	0.95	6117
44	275	7	0.75	0.81	6062
45	500	19	1.03	0.80	6051
46	500	18	0.72	0.67	6048
47	500	18	0.85	0.89	6035
48	270	6	0.96	0.76	6024
49	485	18	0.85	0.88	6023
50	575	18	1.12	0.99	5993
51	495	18	0.85	0.88	5985
52	490	18	0.75	0.81	5971
53	490	18	0.81	0.84	5962
54	490	18	0.85	0.89	5944
55	575	17	0.85	0.88	5940
56	260	6	0.98	0.97	5927
57	480	17	0.85	0.89	5885
58	485	18	1.08	0.77	5882
59	325	6	0.72	0.67	5880
60	485	18	0.82	0.79	5880
61	330	6	0.96	0.96	5876
62	330	6	0.85	0.89	5873
63	485	18	1.09	0.95	5865
64	480	18	0.96	0.76	5863
65	480	18	0.98	0.97	5856
66	460	18	1.05	0.95	5800
67	565	17	1.03	0.80	5791
68	475	19	0.95	0.92	5782
69	250	6	1.00	0.92	5778
70	460	18	0.85	0.89	5775

PŘÍLOHA C: ALGORITMUS ANALYTICKÉHO PROGRAMOVÁNÍ V JAZYCE LUA

```
function analyticalProgramming()
self = {}
    local limitValue = 1000000
    local insert = table.insert
    local constantUpper = 10
    local constantLower = 0

local function range (minimum, maximum)
    local ra = {}
    for i = minimum, maximum do
        insert(ra,i)
    end
    return ra
end

local function toBounds(value, lowerBound,upperBound)
    return lowerBound+value*(upperBound-lowerBound)
end

local function roundTable(candidate)
    local list = {}
    local fl = math.floor
    for i=1,#candidate do
        list[i]=fl(candidate[i])
    end
    return list
end

local function ifIsConstant(num, set, indNum)
    if set[num][1]== "C" then
        return {toBounds(indNum,constantLower,constantUpper),0}
```

```
    else
        return set[num]
    end
end

function self.giveFunction(ind)
    local individual = roundTable(ind)
    local dm = #ind
    local freeposit = dm - 1
    local posit = 1
    local tbl = {}
    local tbl2 = {}
    local choosenFunction = 0
    local temp = {}

    for i=1, dm do
        if freeposit >= 3 then
            choosenFunction = (individual[posit] % #functions)+1
            freeposit = freeposit - functions[choosenFunction][2]
            tbl[i] = ifIsConstant(choosenFunction,
                                  functions,ind[i]-individual[i])

        elseif freeposit == 2 then
            choosenFunction = (individual[posit] % #two_one_zero)+1
            freeposit = freeposit - two_one_zero[choosenFunction][2]
            tbl[i] = ifIsConstant(choosenFunction,
                                  two_one_zero,ind[i]-individual[i])

        elseif freeposit == 1 then
            choosenFunction = (individual[posit] % #one_zero)+1
            freeposit =  freeposit - one_zero[choosenFunction][2]
            tbl[i] = ifIsConstant(choosenFunction,
                                  one_zero,ind[i]-individual[i])
```

```
   elseif freeposit == 0 then
       choosenFunction = (individual[posit] % #zero)+1
       tbl[i] = ifIsConstant(choosenFunction,
                             zero,ind[i]-individual[i])
   end
   posit = posit + 1
end

posit = 1

for i=1, #tbl do
    if tbl[i][2] ~= 0 then
        tbl2[i]={tbl[i][1],range(posit+1,posit+tbl[i][2])}
        posit = posit + tbl[i][2]
    else
        tbl2[i]={tbl[i][1],{0}}
    end
end

for i=dm, 1, -1 do
     if tbl2[i][2][1] > 0  then
          for k,v in pairs(tbl2[i][2]) do
               if k > 1 then
                   insert(temp, ",")
               end
               insert(temp, {tbl2[v][1]})
          end
          tbl2[i]={ {tbl2[i][1], "(" ,temp ,")"} , 0}
     temp = {}
     end
end
return _.flatten({tbl2[1][1]})
```

```
end

local function toEvaluationString(tbl)
     return table.concat(tbl)
end

function self.initialize (f)
     functions = f
     two_one_zero = _.select(functions,
          function(i) return i[2]==0 or i[2]==1 or i[2]==2 end)
     one_zero = _.select(functions,
          function(i) return i[2]==0 or i[2]==1 end)
     zero = _.select(functions,
          function(i) return i[2]==0 end)
end

function self.evalTable(tbl,show)
    local res
    local fnc = loadstring("result =  " .. toEvaluationString(tbl))

    if fnc == nil then
        if show then
             print("Equation : not well formed equation")
        end
        return limitValue
    else
        if show then
            print("Equation : " .. toEvaluationString(tbl))
        end

        fnc()
        res = result
        result = nil
```

```
        if isNAN(res) or isINF(res) then
            return limitValue
        end
        return res
    end
end

return self
end
```