Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Goran Mauša

POBOLJŠANJE POSTUPAKA ZA PREDVIĐANJE PROGRAMSKIH NEISPRAVNOSTI ZASNOVANO NA STROJNOM UČENJU

DOKTORSKI RAD

Zagreb, 2016.

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Goran Mauša

POBOLJŠANJE POSTUPAKA ZA PREDVIĐANJE PROGRAMSKIH NEISPRAVNOSTI ZASNOVANO NA STROJNOM UČENJU

DOKTORSKI RAD

Mentorica: prof. dr. sc. Bojana Dalbelo Bašić
Mentorica: doc. dr. sc. Tihana Galinac Grbac

Zagreb, 2016.

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Goran Mauša

IMPROVEMENT OF SOFTWARE DEFECT PREDICTION METHODS BASED ON MACHINE LEARNING

DOCTORAL THESIS

Supervisor: Professor Bojana Dalbelo Bašić, PhD
Supervisor: Assistant Professor Tihana Galinac Grbac, PhD

Zagreb, 2016

Doktorski rad izrađen je na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Mentorica: prof. dr. sc. Bojana Dalbelo Bašić
Mentorica: doc. dr. sc. Tihana Galinac Grbac

Doktorski rad ima: 171 stranica

Doktorski rad br.: _____

O mentorima

Bojana Dalbelo Bašić rođena je u Zagrebu, 1958. godine. Diplomirala je matematiku na Sveučilištu u Zagrebu Prirodoslovno-matematičkom fakultetu 1982. godine, a magistrirala i doktorirala u polju računarstva na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, 1993. odnosno 1997. godine.

Od 1982. do 1986. radila je u Građevinskom institutu u Zagrebu, od 1986. do 1990. radila je u tvrtki Iskra Delta, a od 1990. do 2000. radila je na Šumarskom fakultetu Sveučilišta u Zagrebu. Od 2000. godine radi na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva. U prosincu 2013. godine izabrana je u trajno zvanje redovitog profesora. U 2000. godini znanstveno se usavršavala na Sveučilištu Erlangen-Nürnberg (DAAD stipendija) i bila je gostujući znanstvenik na institutu INRIA u Rennesu, Francuska, 2005. godine. Bila je voditelj četiri međunarodna i pet domaćih znanstvenoistraživačkih projekta te je bila suradnik na jedanaest drugih projekata. Osnivač je i voditelj Laboratorija za analizu teksta i inženjerstvo znanja na FER-u. Bila je mentor četvorici doktoranda i mentor studentima na 90 diplomskih i završnih radova Objavila je više od 95 radova u časopisima i na znanstvenim konferencijama iz područja obrade prirodnog jezika, pretraživanja informacija, strojnog učenja, inteligentnih sustava i multivarijatne statistike.

Prof. Dalbelo Bašić je član suradnik Hrvatske akademije tehničkih znanosti (HATZ), senior član udruge IEEE, član i suosnivač Hrvatskog društva za jezične tehnologije, član Hrvatskog biometrijskog društva i Hrvatskog matematičkog društva. Sudjelovala je u više međunarodnih programskih odbora međunarodnih znanstvenih konferencija i bila recenzentom u većem broju međunarodnih časopisa i konferencija. Bila je voditelj tima nagrađenog nagradom "Zlatno Teslino jaje" 2007. i 2009. godine te Premijerkinom www nagradom 2009. godine za projekt CADIAL. Za svoj rad, 2015. godine, dobila je priznanje "Zlatna plaketa Josip Lončar" koje dodjeljuje FER.

Tihana Galinac Grbac rođena je u Puli 29. travnja 1977. godine. Diplomirala je, magistrirala i doktorirala u polju elektrotehnike na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 2000., 2005. odnosno 2009. godine.

Od prosinca 2007. zaposlena je na Tehničkom fakultetu Sveučilišta u Rijeci. U listopadu 2010 izabrana je u docenta u polju računarstva. Uspostavila je brojne međunarodne suradnje na području računarstva u sklopu Erasmus, Ceepus i DAAD programa. Ustrojila je Laboratorij za programsko inženjerstvo i obradu informacija – SEIP Lab. Sudjelovala je na dva znanstvena projekata Ministarstva znanosti, obrazovanja i sporta Republike Hrvatske i dva EU COST projekta. Trenutno je voditelj uspostavnog istraživačkog projekta financiranog od Hrvatske zaklade za znanost pod nazivom Evolving Software Systems: Analysis and Innovative Approaches for Smart Management (EVOSOFT). U razdoblju od 2000. do 2007. godine radila je kao softverski

i sistemski inženjer u razvojno-istraživačkom centru Ericsson Nikole Tesle u Zagrebu. Kontinuirano se usavršavala i u drugim razvojno-istraživačkim centrima unutar Ericsson globalne korporacije u Švedskoj, Njemačkoj, Italiji, Grčkoj, Australiji i Kanadi. Suradnja sa Ericsson Nikolom Teslom se nastavila kroz niz projekata u sklopu SEIP Laba. Aktivana je u projektima lokalnog razvoja e-Županija, i Centru kompetencija za pametne gradove te pomaže lokalnim gospodarstvenicima i zajednicama u europskim projektima i prijavama, iUrban, eGov4, te projekta TechLab Grada Rijeke. Objavila je više od 50 znanstvenih radova u međunarodno priznatim časopisima i međunarodnim konferencijama u području programskog inženjerstva, i upravljanja u programskom inženjerstvu.

Doc. Galinac Grbac organizirala je nekoliko međunarodnih znanstvenih skupova i ljetnih škola, član je uređivačkog tima Engineering Review časopisa, uređivala je zbornike znanstvenih skupova, član je programskih i znanstvenih odbora u nizu međunarodnih koneferencija. Recent je u većem broju međunarodnih časopisa. Član je IEEE, ACM i MIPRO.

About the Supervisors

Bojana Dalbelo Bašić was born in Zagreb in 1958. She received her BSc in mathematics from the University of Zagreb, Faculty of Sciences, in 1982 and the MSc and PhD in computer science from the University of Zagreb, Faculty of Electrical Engineering and Computing, in 1993 and 1997, respectively.

From 1982 to 1986 she worked at Civil Engineering Institute, from 1986 to 1990 she worked at Iskra Delta Company, and from 1990 to 2000 she worked at University of Zagreb, Faculty of Forestry. From 2000 she is working at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at FER. In December 2013 she was promoted in Full Professor (permanent position). In 2000 she received DAAD grant for the study visit at the Erlagen-Nürnberg University and in 2005 she was visiting researcher at the institute INRIA, Rennes, France. She coordinated four international projects, five national projects and participated in eleven others research projects. She is head and founder of the Text Analysis and Knowledge Engineering Lab at FER. She supervised four doctoral thesis and 90 diploma, BSc and MSc theses. She published more than 95 papers in the field natural language processing, information retrieval, machine learning, intelligent systems and multivariate statistics.

Prof Dalbelo Bašić is associate member of the Croatian Academy of Engineering, IEEE senior member, member and cofounder of the Croatian Language Technologies Society, member of the Croatian Mathematical Society and the Croatian Biometrical Society. She participated in several international program committees and she was reviewer for a number of international journals and conferences. She was leader of the team awarded by "Golden Tesla's Egg Award" in 2007 and 2009 and by the Prime Minister www Award in 2009 for the CADIAL project. In 2015 she received "Golden Plaque Josip Lončar" from the Faculty of Electrical Engineering and Computing.

Tihana Galinac Grbac was born in Pula in 1977. She received B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 2000, 2005 and 2009, respectively.

From December 2007 she is working at the Faculty of Engineering University of Rijeka. In October 2010 she was promoted to Assistant Professor in Computer Science. She established number of international collaborations in Computer Science within Erasmus, Ceepus and DAAD programs. She established Software Engineering and Information Processing Laboratory - SEIP Lab. She participated in 2 scientific projects financed by the Ministry of Science, Education and Sports of the Republic of Croatia and 2 EU COST projects. Currently she is a project leader of the installation research project: "Evolving Software Systems: Analysis and Innovative Approaches for Smart Management (EVOSOFT)" financed by the Croatian Science Foundation. In period from 2000. until 2007. She was working as software and system engineer

in research and development center of Ericsson Nikola Tesla in Zagreb. She was professionally specialized in other research and development centers within Ericsson global corporation: Sweden, Germany, Italy, Greece, Australia, Canada. Collaboration with Ericsson Nikola Tesla is continued through number of projects realized within SEIP Lab. She is active in local development projects e-Županija, i Competence Center for Smart Cities and supports local bodies and communities in European projects and applications, iUrban i eGov4, and TechLab, projects of City of Rijeka. She published more than 50 papers in journals and conference proceedings in the area of software engineering and software engineering management.

Assist. Prof. Galinac Grbac organized several international scientific conferences and summer schools. She is associate editor in Engineering Review journal and was editing SQAMIA conference proceedings, and is a member program and scientific boards in number of international conferences. She serves as a technical reviewer for various international journals. She is a member of IEEE, ACM and MIPRO.

Praeterita magis reprehendi possunt quoam corrigi.
(Prošlost se više ne može kuditi nego samo ispraviti. LIV XXX, 30, 7.)

Ratio praeteriti scire futura facit.
(Tumačenje prošlosti omogućuje znanje o budućnosti.)

...mojoj majci...

Veliko hvala obitelji, prijateljima, kolegama i suradnicima za svu podršku, a posebno:

- ◇ Tihani Galinac Grbac – mojoj mentorici sa Tehničkog fakulteta, koja se od samog početka bezrezervno posvetila stručnom vođenju moga znanstvenog puta. Uvijek je bila spremna pomoći, motivirati i prijateljski savjetovati. Uistinu se smatram sretnim što sam imao takvu mentoricu.

- ◇ Bojani Dalbelo Bašić – mojoj mentorici sa FER-a, koja je uvijek bila spremna zaobići sve poteškoće i pomoći na svaki mogući način kako bi se moj doktorski studij uspješno i pravovremeno završio.

- ◇ Vladi Sruku, Željki Car i Ivi Ipšiću – članovima povjerenstva za ocjenu i obranu doktorskog rada, koji su detaljno proučili moju disertaciju te ju unaprijedili vrijednim savjetima i ispravkama.

- ◇ Sandiju Ljubiću i Damiru Arbuli – kolegama i dragim prijateljima iz ureda, za vječiti smijeh, dobro društvo i ugodnu atmosferu zbog kojih sam naučio što znači zdrava radna atmosfera. Bez našeg ritualnog ispijanja čaja te zafrkancije ne bi bio toliki gušt ući u ured i reći "doktors".

- ◇ Tati i Mladenu – mojoj obitelji, koja me podupire u svim mojim planovima, kojima ništa nije teško učiniti za mene i koji su vjerojatno ponosniji na ovaj doktorat čak i od mene samog.

- ◇ Iri – meni najdražoj osobi s kojom je svaki trenutak savršen, koja me uvijek može oraspoložiti i koja mi u svakoj situaciji pruža osjećaj da pored sebe imam "nekog svog".

Sažetak

Programski sustavi današnjice kontinuirano se unapređuju i evoluiraju. Povećanje složenosti programskih sustava uzrokuje povećanje opsega verifikacijskog djelovanja, a time i troškova razvoja. Programske neispravnosti nejednoliko su raspoređene po programskim sustavima i to na način da se u manjem postotku programa nalazi veći postotak neispravnosti. Zbog toga, ovaj rad bavi se predviđanjem dijelova sustava sa programskim neispravnostima s ciljem pametnog usmjeravanja verifikacijskih strategija. U ovom radu izložen je sustavni pristup predviđanju programskih neispravnosti. Postupci prikupljanja podataka za potrebe predviđanja programskih neispravnosti nisu normirani i osnovni su uzrok nemogućnosti poopćenja primjene metoda predviđanja. Zbog toga, u radu je definiran postupak za prikupljanje podataka za potrebe predviđanja programskih neispravnosti s ciljem povećanja prikladnosti podataka za primjenu metoda predviđanja. Postupak se temelji na postojećim industrijskim normama koje definiraju atribute za sustavno praćenje neispravnosti programskih sustava u organizacijama razvoja i skupu statičkih metrika za objektno orijentirani programski kod. Drugi problem kojem se posvećuje rad je istraživanje mogućnosti primjene metoda strojnog učenja na neujednačenim skupovima podataka. Taj problem poznat je u literaturi strojnog učenja, a podaci o programskim neispravnostima su inherentno neujednačeni. U radu je definirana metoda za utvrđivanje granične razine neujednačenosti za izgradnju prikladnog modela predviđanja programskih neispravnosti zasnovanog na metodama strojnog učenja. Uz metodu predložen je postupak i verifikacija odabira kombinacije metoda za predviđanje programskih neispravnosti iz skupa metoda za pripremu podataka i metoda za strojno učenje. Pomoću predloženih postupaka moguće je unaprijed bolje procijeniti neispravne dijelove programskog sustava i time poboljšati verifikacijsku i razvojnu strategiju programskih sustava te unaprijediti planiranje budućih ulaganja u razvoj složenih programskih sustava u evoluciji. Svi ostvareni rezultati provjereni su iscrpnim studijama slučaja i sustavim pregledom literature, osnovnim metodologijama definiranim za primjenu u disciplini programskog inženjerstva.

Ključne riječi: predviđanje programskih neispravnosti, prikupljanje podataka, neujednačeni skupovi podataka, strojno učenje

Extended Abstract

Improvement of Software Defect Prediction Methods Using Machine Learning

The complexity of modern software is constantly rising and its development is being constrained with strict deadlines and limited budget. Consequently, there is an increasing need for fast and accurate software quality assurance that could be used as early as possible. There are many aspects of quality, but one of the most important ones is the number of failures that are discovered during testing. To minimize their occurrence, verification and validation activities are being conducted during the whole life cycle of software development. Testing the software code is one of the verification activities that requires a lot of resources and desperately needs strategies that would make it more efficient. This thesis is exploring the strategy of software defect prediction (SDP) that could detect parts of software code that is more prone to defects, focus testing resources and reduce costs. The IEEE 1012 norm suggests minimal demands for development of verification and validation plans. This thesis is related to the parts of this norm that are concerned with the analysis of project inputs because SDP requires historical data. It is also related to the choice of proper techniques and tools because the data needs to be collected in a consistent manner and because there is a great number of possible prediction models that could be used for SDP.

Thesis Goals

The current state in SDP research is in need of a systematically defined data collection procedure. There are only the most often used procedures and the quality of data is being heavily criticized. The data collection procedure is a difficult and a time consuming task and few research groups attempt to do it. That is why the first goal of this thesis is to establish systematically defined guidelines for unbiased data collection. The expected scientific contributions are (1) the data collection procedure for SDP that will increase the appropriateness of data, and (2) the data collection algorithm based on static code attributes for unstructured and formally unlinked repositories of software code and faults. Another problem in SDP is the lack of guidelines for practitioners about which prediction model should they use for their verification and validation plans. There are many machine learning algorithms that obtain good results in SDP, but it is unrealistic to expect one technique that is always the best. Instead, this thesis seeks to find a parameter which describes the context of the application of SDP and which contains boundaries for more specific guidelines. The chosen parameter is the level of data imbalance, because it is an inherent property of SDP data and because it is known that high levels of data imbalance deteriorate machine learning capabilities. In other words, the goal of this thesis is to determine at which level of data imbalance do the machine learning algorithms become incapable of performing their prediction task and whether there are specific levels of data imbalance within

which certain machine learning techniques yield consistently better results that the others. The expected scientific contributions are (1) the method for establishing the critical level of data imbalance for training the SDP model, and (2) the procedure and verification of choosing the appropriate combination of data preprocessing and machine learning methods for SDP.

Thesis Structure

The research questions that drive this thesis are in its briefest form: (1) how to perform data collection? and (2) how to build a successful prediction model? The methodology that was used to give answers to those questions are: a systematic literature review, an investigation of data collection parameters, a comprehensive comparative study for establishing the most appropriate data collection procedure, and an empirical case study that investigates the impact of different levels of data imbalance on machine learning based predictive models. The thesis consists of the following sections.

Section 1 gives an introduction to the research field and to the background and the importance of SDP. It also presents the goals of this research, the problems are going to be examined and the methodology to do it.

Section 2 describes in details the underlying problems of the research questions that are in focus of this thesis. It describes the potential sources of data for SDP, the popular data collection procedures and the techniques for linking the repositories that do not have a formal link. It also describes the general process of building the prediction model, the problem of data imbalance and the machine learning algorithms that are used in this thesis.

Section 3 presents the results of the systematic literature review. It presents all the data collection parameters that were found in related work, like the defect severity, the software development repositories' search order or like what should be done with duplicated data. This section also contains an exploratory case study which aims to find the most suitable tools for collecting the static code attributes and to examine in practice whether there are additional data collection parameters that were not mentioned in related work.

Section 4 is based on the results obtained in previous section. It presents the data collection tool that implements the systematically defined data collection procedure and a new repository linking technique. It also contains a case study which analyses the developed tool and technique and which yields some very promising results.

Section 5 completes the data collection topic of this thesis with a comprehensive comparative case study of known and available data collection procedures and repository linking techniques. The importance of this case study is also in giving a framework for future systematic comparisons of data collection procedures and linking techniques. The results revealed that the proposed data collection procedure and algorithm yield the most accurate data for SDP.

Section 6 examines the problem of unbalanced datasets and whether it can be used to describe

the context of application of machine learning algorithms in SDP. The method for establishing the critical level of data imbalance for training the SDP model is proposed in this section. The critical level of data imbalance is the ratio of minority class in a dataset below which the performance of machine learning algorithm starts to deteriorate steeply. The method for establishing the most successful machine learning technique with regards to the present level of data imbalance is also proposed in this section. This section also contains an empirical case study that investigates the impact of different levels of data imbalance on machine learning based predictive models. The results of this empirical case study revealed that the proposed methods successfully complete the tasks they were designed for and that this thesis achieved the second two scientific contributions.
Section 7 contains two appendices. The first appendix describes the object-oriented static code attributes that are collected with the developed tool. These metrics are the independent variables in SDP and the basis of the case study that is performed in section 6. The second appendix contains the details of statistical analysis and tests that are conducted in section 6.

Conclusion

This thesis was focused on two major problems in SDP. The first problem is the inability to compare and generalize the results of past research. The major reason for this is the insufficient construction validity, i.e. the problem with data quality. The lack of standards presents the greatest challenge in the data collection process. This thesis managed to include the existing standards into a systematically defined data collection procedure. It also proposed a repository linking technique that is based on regular expressions, which outperforms other known techniques. These contributions were implemented in a data collection tool so that future research could be performed on a unified and unbiased basis. The second problem is that SDP research community invested a lot of effort into finding the optimal machine learning technique, but with limited success. This thesis proposed the usage of level data imbalance as a description of the context of application of SDP. The method of establishing the critical level of data imbalance that is based on Arrow-Pratt metric enables us to determine above which level of imbalance certain machine learning algorithms become incapable of performing their defect prediction task. The other proposed method enables the practitioners to find the most appropriate machine learning method for the level of imbalance that they are facing in practice. The improvement of SDP methods that is presented in this thesis will increase the quality of research in this field and it will enhance the software code verification strategy in the earliest stages of development of complex and evolving software.

Keywords: software defect prediction, data collectio, data imbalance, machine learning

Sadržaj

Popis slika

Popis tablica

Popis kratica

%SNP	udio programskih modula Sklonih Neispravnostima	uvedena kratica
A(%SNP)	Arrow-Prattova mjera u ovisnosti o udjelu *%SNP*	uvedena kratica
ACC	točnost, eng. *Accuracy*	uobičajena kratica
ACM	eng. *Association for Computing Machinery*	ime organizacije
ANOVA	analiza varijance, eng. *Analysis Of Variance*	uobičajena kratica
AUC	površina ispod krivulje, eng. *Area Under Curve*	uobičajena kratica
Ave	težinska prosječna točnost, eng. *weighted Average accuracy*	uobičajena kratica
BD_i	Broj povezanih Datoteka (na izlazu iz tehnike povezivanja)	uvedena kratica
BD_u	Broj Datoteka na ulazu u tehniku povezivanja	uvedena kratica
BIRT	eng. *Business Intelligence and Reporting Tools*	ime projekta
BN_i	Broj povezanih Neispravnosti (na izlazu iz tehnike povezivanja)	uvedena kratica
BN_u	Broj Neispravnosti na ulazu u tehniku povezivanja	uvedena kratica
BP_i	Broj povezanih Predaja (na izlazu iz tehnike povezivanja)	uvedena kratica
BP_u	Broj Predaja na ulazu u tehniku povezivanja	uvedena kratica
BuCo	Bug-Code Analyzer	ime alata
BV_i	Broj uspostavljenih Veza (na ilazu iz tehnike povezivanja)	uvedena kratica
DN	uređeni par (Datoteka, Neispravnost)	uvedena kratica
FM	F-mjera, eng. *F-Measure*	uobičajena kratica
FN	eng. *False Negative*	uobičajena kratica
FP	eng. *False Positive*	uobičajena kratica
FPR	udio netočno pozitivnih modula, eng. *False Positive Rate*	uobičajena kratica
GM	geometrijska sredina, eng. *Geometric Mean*	uobičajena kratica
GVN	Granična Vrijednost Neispravnosti	uvedena kratica
HTTPD	eng. *Hypertext Transfer Protocol*	uobičajena kratica
IEC	eng. *International Electrotechnical Commission*	ime organizacije
IEEE	eng. *Institute of Electrical and Electronics Engineers*	ime organizacije
ID	identifikacijska oznaka, eng. *Identifier*	uobičajena kratica
ISO	eng. *International Organization for Standardization*	ime organizacije
IT	eng. *Information Technology*	industrijska grana
JDT	eng. *Java Development Tools*	ime projekta

maks	najveća vrijednost u skupu podataka	uvedena kratica
MCC	Mathew koeficijent korelacije, eng. *Mathews Correlation Coefficient*	uobičajena kratica
min	najmanja vrijednost u skupu podataka	uvedena kratica
NLP	eng. *Natural Language Processing*	uobičajena kratica
NNL	eng. *Natural Log Likelihood*	uobičajena kratica
NNP	Nesklonost Neispravnostima	uvedena kratica
Os	Osjetljivost	uobičajena kratica
PCA	eng. *Principal Component Analysis*	uobičajena kratica
PDE	eng. *Plug-in Development Environment*	ime projekta
PR	preciznost, eng. *Precision*	uvedena kratica
PROMISE	eng. *Prediction Models In Software Engineering*	ime repozitorija
Regex	pretraga regularnim izrazima, eng. *Regular expression*	uvedena kratica
ROC	eng. *Receiver Operating Curve*	uobičajena kratica
SDP	eng. *Software Defect Prediction*	uobičajena kratica
SN_i	Suma Neispravnosti u datotekama (na izlazu iz tehnike povezivanja)	uvedena kratica
SNP	Sklonost Neispravnostima	uvedena kratica
SP	Stopa Povezivanja neispravnosti	uvedena kratica
SQL	eng. *Structured Query Language*	uobičajena kratica
sv	srednja vrijednost	uvedena kratica
SVM	eng. *Support Vector Machine*	uobičajena kratica
SVN	eng. *Subversion*	ime sustava
SZZ	Sliwerski, Zimmermann, Zeller	uobičajena kratica
TN	Točno Negativan klasificirani slučaj	uobičajena kratica
TNR	udio točno negativnih modula, eng. *True Negative Rate*	uobičajena kratica
TP	Točno Pozitivan klasificirani slučaj	uobičajena kratica
TPR	udio točno pozitivnih modula, eng. *True Positive Rate*	uobičajena kratica
UML	eng. *Unified Modeling Language*	uobičajena kratica
V&A	kombinacija tehnika Vremenska korelacija i podudaranje Autorstva	uvedena kratica

Poglavlje 1

Uvod

Suvremeni sustavi obavljaju sve složenije funkcije pa se time broj linija koda i složenost programskog koda stalno povećava. S porastom složenosti programskog koda povećava se broj mogućih različitih izvedbi koda pa su time potrebe za verifikacijskim i validacijskim aktivnostima u stalnom porastu i sve je manje moguće provesti verifikaciju i validaciju programskih sustava kojom bi se garantirala određena pouzdanost programskog sustava. Udio programskog sustava koji je moguće verificirati u realnim industrijskim uvjetima, uslijed vremenskih i resursnih ograničenja, sve se više smanjuje. Kao posljedica takvog stanja dolazi do smanjenja pouzdanosti isporučenih programskih sustava te povećanja broja neispravnosti i troškova održavanja takvih sustava. Pojedine studije procjenjuju da se u verifikaciju i validaciju ulaže i do 80% ukupnih troškova nakon isporuke proizvoda, što se nepovoljno odražava na poslovanje organizacija razvoja takvih sustava [1]. Časopis *World Quality Report 2015*[*] navodi da se 35% ukupnih troškova u IT industriji u 2015. godini odnosilo na osiguranje kvalitete i da će taj trošak narasti na 40% ukupnih troškova razvoja do 2018. godine. Zaključno, vodeća svjetska istraživanja u disciplini programskog inženjerstva usmjerena su upravo prema tom industrijskom problemu.

Jedan od važnijih smjerova istraživanja jesu ideje poboljšanja strategije razvoja uključivanjem kontinuirane i automatizirane verifikacije i validacije. U potrazi za općim principima ponašanja neispravnosti u programskim sustavima koji bi se mogli formalizirati i automatizirati, pokrenut je čitav niz empirijskih istraživanja. U stvarnoj okolini industrijskog razvoja složenih programskih sustava došlo se do nekoliko važnih zaključaka. Neispravnosti su nejednoliko raspodijeljene po modulima složenih industrijskih programskih sustava. Moduli s većim udje-

[*] http://www.computerweekly.com

lom ukupnih neispravnosti ne predstavljaju veći udio veličine koda programskih sustava [2–4]. Upravo ta spoznaja motivirala je cijelu grupu istraživanja unutar discipline programskog inženjerstva pod nazivom *predviđanje programskih neispravnosti* (eng. Software Defect Prediction, SDP). U velikim i složenim programskim sustavima važnu ulogu u definiranju strategije razvoja imaju analitička predviđanja programskih neispravnosti u svrhu usmjeravanja budućeg razvoja, verifikacije i validacije te usavršavanja kompetencija timova za provedbu naprednijih načina testiranja.

Programsko inženjerstvo je relativno mlada inženjerska grana, koja nastoji uspostaviti pravila u procesu razvoja programskih sustava koja su zasnovana na dobrim i sustavno prikupljenim iskustvima. Disciplina je definirana s ciljem rješavanja problema loše kvalitete programskih sustava, prekoračenja vremenskih ograničenja i troškova te uvođenja sustavnog, strogog i mjerljivog procesa koji bi se odvijao unutar zadanog vremena, budžeta i u skladu sa zahtjevima [5]. Do sada prikupljene industrijske prakse nisu pokazale sustavnost u postupcima za predviđanje programskih neispravnosti. Ova disertacija se bavi problemom poboljšanja postupka predviđanja neispravnosti zasnovano na strojnom učenju koje pomažu procesima definiranja strategije razvoja složenih programskih sustava.

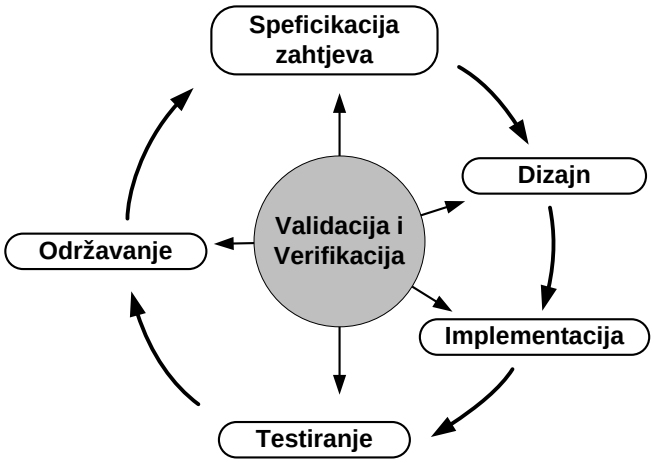
1.1 Područje istraživanja

Jedan od zadataka osiguranja kvalitete programskih sustava je rani i učinkoviti pronalazak neispravnosti. Ova disciplina vođena je normama koje priznaju nacionalne ili internacionalne organizacije, industrijski čimbenici ili vladina tijela. Norme preslikavaju aktualni korpus znanja koji pruža temelj za razvoj profesionalnih disciplina. Poznato je da pronalazak neispravnosti u ranijim fazama razvoja programskih sustava smanjuje troškove, a zanemarivanje osiguranja kvalitete u ranim fazama rezultira značajno većim brojem neispravnosti u kasnijoj fazi testiranja i održavanja, što uzrokuje veće troškove ispravljanja neispravnosti [6].

Kako bi se smanjili troškovi, izuzetno je značajno definirati sustavne postupke koji će osigurati pronalazak što većeg broja neispravnosti u što ranijoj fazi razvoja. Iako je ovo dobro poznata činjenica, industrija i dalje troši značajna sredstva u ispravljanje neispravnosti u kasnijim fazama [7]. Jedna od glavnih prepreka u provođenju opsežnih aktivnosti osiguranja kvalitete je vremenski pritisak koji požuruje sve faze razvoja. Taj pritisak dodatno raste s porastom složenosti novih funkcionalnosti i programskih sustava. Empirijska istraživanja pokazala su da se velik postotak neispravnosti grupira u određenim programskim modulima koji, po broju linija koda, predstavljaju manji dio cjelovitog programskog sustava [2–4]. Upravo takva, empirijski dokazana, distribucija neispravnosti motivira istraživanja čiji je cilj pronalazak analitičkog modela pomoću kojeg bi se moglo otkriti problematične module u strukturi programskog sustava. Ova disertacija orijentirana je na unapređenje postupaka za pronalazak odgovarajućeg analitičkog modela koji bi pomogao u donošenju strategije verifikacije složenih programskih sustava u evoluciji.

Nazivlje

Prema normi ISO/IEC 12207, proces životnog ciklusa programskog sustava može se podijeliti u čak 23 procesa, 95 aktivnosti, 325 zadataka i 224 artefakta [8]. Osnovne faze u procesu razvoja programskog sustava tradicionalno se dijele na inženjerstvo zahtjeva, dizajn rješenja, implementaciju, testiranje i održavanje [9]. Redoslijed izvođenja ovih faza definiran je odabranim modelom razvoja koji se zbog njihove međuovisnosti najčešće izvodi iterativno. Inženjerstvo zahtjeva prvi je korak svakog modela razvoja, a cilj mu je opis problema koji se rješava i definicija zahtjeva. Dizajn rješenja podrazumijeva dekompoziciju problema na manje programske module koji će zadovoljiti prethodno postavljene zahtjeve. Implementacija je faza programiranja programskih modula koji sačinjavaju cjelovit sustav. Testiranje programskog koda je faza



Slika 1.1: Faze u procesu razvoja programskog sustava

koja je u uskoj sprezi s fazom implementacije, i poželjno je što ranije započeti s njezinom provedbom. Održavanje je konačna faza, čija je svrha otkloniti neispravnosti nakon isporuke te prilagođavati i poboljšavati sustav promjenjivim zahtjevima. U razvoju dugovječnih projekata, čija evolucija podrazumijeva velik broj inačica, održavanje nije ujedno i posljednja faza. Ono što je zajedničko svim navedenim fazama je provedba verifikacije i validacije, kao što je to prikazano slikom 1.1.

Verifikaciju i validaciju važno je razlikovati od faze testiranja programskog koda jer se one mogu odnositi i na dokumentaciju i druge artefakte u procesu razvoja. Proces *verifikacije* (eng. verification) je proces provjere ili procjenjivanja programskog sustava i pripadajuće dokumentacije u ispunjenju zahtjeva i/ili uvjeta zadanih na početku te faze. Proces *validacije* (eng. validation) je proces provjere ili ocjenjivanja programskog sustava i pripadajuće dokumentacije u ispunjenju specificiranih zahtjeva. Primjerice, u testiranju programskog koda - verifikacija provjerava sadrži li programski sustav sve tražene module i rade li oni ispravno, a validacija provjerava hoće li sustav zadovoljiti korisnikove potrebe [9]. S obzirom na navedenu podjelu faza razvoja programskog sustava, u središtu ove disertacije jest donošenje strategije verifikacije i validacije programskog koda.

Provedba verifikacije i validacije je sastavni dio osiguranja kvalitete. Postoje brojni aspekti na koje se osiguranje kvalitete može usmjeriti, ali najočitiji indikator nedovoljne razine kvalitete jesu neispravnosti koje se prijavljuju prilikom testiranja ili uporabe proizvoda. U programskom inženjerstvu postoje brojni nazivi koji se mogu pogrešno protumačiti kao sinonimi za neispravnosti. U ovoj disertaciji koristi se nazivlje iz norme IEEE 1044-2009 [10]:

- *Oštećenje* (eng. defect): nesavršenost ili nedostatak u proizvodu koji radi, ali ne zadovo-

ljava zahtjeve ili specifikaciju te ga je potrebno popraviti ili zamijeniti;

- *Pogreška* (eng. error): neispravno uneseni podatak, korak, proces ili instrukcija u izvornom kodu ili pripadajućem dokumentu, uzrokovan ljudskom nepažnjom;
- *Kvar* (eng. failure): prekid sposobnosti zadovoljavanja očekivanih funkcija ili nemogućnost rada proizvoda u prethodno definiranim granicama;
- *Neispravnost* (eng. fault, bug): manifestacija pogreške pri izvođenju programa.

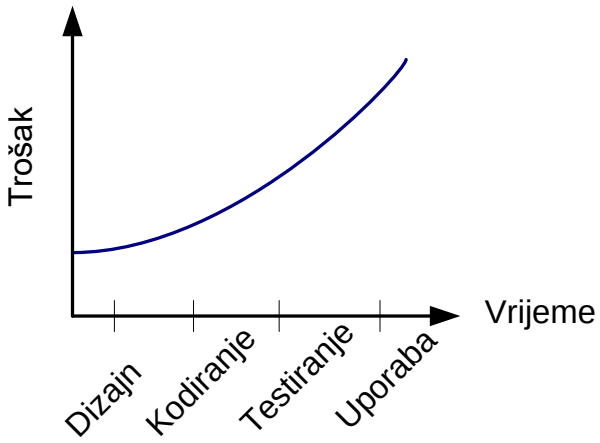
Neispravnost u načelu može biti unutar programskog sustava ili u potpornim sustavima u okruženju, uključujući i projektnu dokumentaciju. Ako se neispravnost dogodi prilikom izvođenja programskog sustava, može uzrokovati jedan ili više kvarova. Uobičajeni slijed događaja u testiranju ili uporabi programskog sustava je sljedeći:

1. U razvoju se učini pogreška.
2. Pogreška se manifestira kao neispravnost koja uzrokuje jedan ili više kvarova prilikom testiranja.
3. Kvarovi se prijavljuju na način da ih je moguće ponoviti.
4. Nakon što se prijavljeni kvarovi analiziraju, otkriva se neispravnost koja ih je uzrokovala i tada je moguće ispraviti pogrešku koja je nastala ljudskom nepažnjom za vrijeme razvoja.

Pogreške su neizbježne jer ljudi pišu programski kod i popratnu dokumentaciju pa ih zato nije moguće predvidjeti. Neispravnosti su manifestacije tih pogrešaka, a njih nastojimo predvidjeti kako bismo uklonili oštećenja i kvarove. Stoga se u engleskom nazivlju mogu pronaći nazivi *software defect prediction* i *software fault prediction*, koji su istoznačnice, a u ovoj se disertaciji prevode kao predviđanje programskih neispravnosti. Engleski naziv *bug*, koji se često koristi u svakodnevnom govoru, nema strogu definiciju. Budući da poznati sustav za praćenje zahtjeva Bugzilla, koji je korišten i u ovoj disertaciji, omogućuje praćenje neispravnosti (onih koji uzrokuju i kvarove i oštećenja), pojam *bug* bi najispravnije bilo prevesti kao neispravnost.

Troškovi vezani uz testiranje

Kvaliteta podrazumijeva troškove u razvoju bez obzira na odabranu strategiju osiguranja kvalitete. Strategija visokog stupnja kvalitete ulaže značajna sredstva u osiguranje da razvijeni sustav nema neispravnosti, a odabirom strategije niskog stupnja kvalitete dolazi do naknadnih troškova zbog kvarova. Unapređenje učinkovitosti i smanjenje tih troškova moguće je zbog razlike u cijeni ispravljanja neispravnosti u različitim fazama razvoja, kao što je prikazano slikom 1.2. Neka istraživanja navode da je cijena pronalaska i ispravljanja neispravnosti često



Slika 1.2: Trošak neispravnosti u fazama razvoja programskog sustava

100 puta veća nakon isporuke programskog sustava korisnicima, nego za vrijeme faze dizajna u razvoju [7]. To implicira da je najučinkovitiji pristup onaj koji ulaže najveći napor za pronalazak neispravnosti u najranijim fazama razvoja. Međutim, moduli programskog sustava se ne razvijaju izolirano te je neophodno provoditi testiranje u nekoliko razina: testiranje modula, testiranje funkcionalnosti, integracijsko testiranje i *testiranje prihvaćanja* (eng. acceptance testing). Svaka iduća razina testiranja provodi se u sve kasnijoj fazi razvoja i neophodno ulazi u razrede većeg troška.

U analizi troškova testiranja važno je razlikovati *učinkovitost* (eng. efficiency) i *djelotvornost* (eng. effectiveness) testiranja. Učinkovitost se tradicionalno mjeri kao omjer broja neispravnosti pronađenih u testu i napora koji iziskuje provedba tog testa, dok se djelotvornost mjeri kao postotak pronađenih neispravnosti u ukupnom broju neispravnosti, bez obzira na cijenu njihova pronalaska [11]. Učinkovitost testiranja ovisi o brojnim čimbenicima kao što su: primjerena tehnika za dizajn i odabir testnih slučajeva, dostupnost potpore alata i ispravan redoslijed provedbe testiranja. Učinkovito testiranje analizira sve aspekte programskog sustava prije nego dođe u fazu uporabe, i izbjegava redundantne aktivnosti. Analitički modeli predviđanja, koji su predmet istraživanja ove disertacije, prvenstveno mogu unaprijediti učinkovitost testiranja. Uspješno predviđanje bi omogućilo da se neispravnosti pronalaze već u fazi implementacije - kada je trošak ispravljanja manji. Također, ono bi omogućilo usmjeravanje većih sredstava na programske module koji sadržavaju većinu neispravnosti. Na taj način bi se, posljedično, unaprijedila i djelotvornost testiranja. Budući da se testiranje provodi unutar ograničenog vremenskog i financijskog okvira, iluzorno je očekivati apsolutnu djelotvornost. No, ovim načinom mogu se očekivati ispravljanja većeg postotka neispravnosti.

Provedba testiranja

Međunarodna norma za testiranje programskih sustava ISO/IEC/IEEE 29119 stručnjacima za osiguranje kvalitete pruža smjernice za provedbu testiranja u čitavom životnom ciklusu razvoja. Tom su normom definirani koncepti i pojmovi, procesi testiranja, testna dokumentacija i tehnike testiranja [12]. Fokus testiranja može biti orijentiran na validaciju, pri čemu se provjeravaju funkcije sustava (princip crne kutije), ili na verifikaciju, pri čemu se provjerava struktura programskog sustava (princip bijele kutije) [13, 14]. Ova disertacija usmjerena je na problem verifikacije i donošenja strategije njezine provedbe za složene programske sustave.

Za uspješnu provedbu testiranja, potrebno je temeljito planiranje procesa i odabir odgovarajućih tehnika. Planiranje testiranja je slojevit i zahtjevan zadatak koji uključuje definiranje ciljeva testiranja, fokus testiranja i dodjelu sredstava [13]. Tek nakon što je općenita strategija verifikacije usvojena može se planirati dodjela sredstava i odabir tehnika za njezinu provedbu. Planiranje uvelike ovisi o dostupnim ljudskim resursima, razini njihova iskustva i znanja. Važno je dobro procijeniti domenu primjene jer različite tehnike mogu biti prikladne i primjenjive samo na određeni tip problema [13].

Razine provedbe testiranja mogu biti *testiranje modula* (eng. component testing), *integracijsko testiranje* (eng. integration testing), *testiranje sustava* (eng. system testing) i *testiranje prihvaćanja* (eng. acceptance testing). Testiranje modula provjerava njihovu logičku ispravnost, integracijsko testiranje provjerava jesu li zadovoljeni zahtjevi dizajna, testiranje sustava provjerava jesu li zadovoljeni zahtjevi za funkcionalnosti, a testiranje prihvaćanja provjerava zadovoljstvo naručitelja programskog sustava [14].

Postoje brojne tehnike koje se mogu koristiti u provedbi testiranja, ali one nisu predmet istraživanja ove disertacije. Možemo ih grubo podijeliti na tehnike koje se provode ručno i one koje se odnose na provedbu testnih slučajeva [9]. Ručne tehnike uključuju *čitanje* (eng. reading) dokumentacije ili programskog koda, *inspekcije* (eng. inspections) od strane nepristranih osoba koje nisu autori dokumenta ili programskog koda i sl. Tehnike koje se odnose na provedbu testnih slučajeva dijele se na testiranje zasnovano na: *pokrivenosti testom* (eng. coverage-based), *neispravnostima* (eng. fault-based) i *pogreškama* (eng. error-based) [9]. Testiranje zasnovano na pokrivenosti testom promatra koji i koliki dio strukture programskog koda ili toka podataka pokrivaju testni slučajevi. Testiranje zasnovano na neispravnostima nastoji pronaći podskup testnih slučajeva koji će otkriti što veći broj neispravnosti. Procjena testnih slučajeva najčešće se radi namjernim *unošenjem pogrešaka* (eng. error seeding) u programski kod, nakon čega se

traži koji testni slučajevi imaju sposobnost otkrivanja tih neispravnosti. Testiranje zasnovano na pogreškama orijentira se na module unutar programskog sustava koji uobičajeno sadrže pogreške. S obzirom na navedene podjele testiranja, tema ove disertacije jest najbliža testiranju programskih modula zasnovanom na pogreškama.

Plan i strategija verifikacije

Kao i sve ostale faze u razvoju programskog sustava, aktivnosti verifikacije moraju biti pomno isplanirane. Precizan opis aktivnosti, zaduženja i procedura mora biti donesen u ranim fazama razvoja kako bi se pogreške što prije otkrile i ispravile, te kako bi se rizici, troškovi i izmjene planova smanjili, a kvaliteta i pouzdanost programskog sustava povećala. Norma IEEE 1012 predlaže minimalne zahtjeve za formu i sadržaj plana verifikacije i validacije, koji su uniformni za programske sustave visokih ili niskih zahtjeva za kvalitetom [14]. Tom normom može se pristupiti planiranju verifikacije i validacije bilo koje faze razvoja programskog sustava. Cilj planiranih aktivnosti može biti da se verificira: (1) jesu li zadovoljeni zadani ciljevi poput ispravnosti, cjelovitosti, konzistencije i točnosti, (2) pridržava li se proces razvoja propisanih normi i konvencija, (3) jesu li ustanovljeni kriteriji prijelaza u iduću fazu razvoja, te da se validira zadovoljava li krajnji produkt tražene zahtjeve. Sadržaj dokumenta plana verifikacije i validacije, sukladno normi IEEE 1012, sadrži sljedeća poglavlja:

1. Svrha
2. Referentni dokumenti
3. Definicije
4. Pregled verifikacije i validacije
   - 4.1. Organizacija
   - 4.2. Glavni raspored miljokaza
   - 4.3. Sažetak raspoloživih resursa
   - 4.4. Podjela zaduženja
   - 4.5. Alati, tehnike i metodologije
5. Proces verifikacije i validacije
   - 5.1. Vođenje
   - 5.2. Faza koncepta
   - 5.3. Faza specifikacije zahtjeva
   - 5.4. Faza dizajna

5.5. Faza testiranja

5.6. Faza održavanja

6. Podnošenje izvještaja
7. Procedura administracije

Važan dio poglavlja 5.1. posvećen je donošenju plana verifikacije i validacije svih faza razvoja programskog sustava. Smjernice za planiranje strategije verifikacije teže njezinoj efikasnoj i kontroliranoj provedbi, a dijele se na sljedeće korake [14]:

1. Identifikacija polja djelovanja - uključuje definiciju zadataka u verifikaciji i metodologiju donošenja odluka na temelju njezinih rezultata;
2. Uspostava specifičnih ciljeva - uključuje detaljan opis mjerljivih i dostižnih uvjeta koji vode ka ostvarenju zadanih ciljeva;
3. Analiza projektnih ulaza - uključuje procjenu svih produkata i informacija koje će biti dostupne za vrijeme trajanja projekta;
4. Odabir tehnika i alata - koje su odgovarajuće s obzirom na projektne ulaze, ustanovljene prethodnim korakom;
5. Definicija plana - uključuje pripremu protokola i skupa zadataka čija će provedba osigurati ostvarenje ciljeva verifikacije.

Upravo u analizi projektnih ulaza planira se analiza i predviđanje temeljeno na povijesnim podacima iz prijašnjih inačica. Podaci o neispravnostima pružaju podlogu za analizu anomalija, planiranje potrebnih resursa za verifikaciju i pomoć u odabiru odgovarajućih tehnika za njezinu provedbu [14]. Ova disertacija nadovezuje se na 3. korak (analizu projektnih ulaza) u vidu prikupljanja podataka iz razvojnih repozitorija sustava za upravljanje inačicama i sustava za praćenje zahtjeva, te na 4. korak (odabir odgovarajućih tehnika i alata) za predviđanje programskih neispravnosti.

Velik dio poglavlja 5.3. i 5.4. odnosi se na planiranje faze verifikacije programskog koda. Pored podjele zaduženja, procjene potrebnih testnih slučajeva i definicije kriterija koje verifikacija programskog koda mora zadovoljiti, plan bi trebao identificirati rizična područja i dodijeliti im resurse. Taj dio plana je u središtu ove disertacije. Za velike i složene sustave važno je odrediti prioritet u provedbi testnih slučajeva. To se prvenstveno odnosi na odabir programskih modula u koje će biti uložena veća, odnosno manja sredstva verifikacije. Postoji nekoliko strategija verifikacije koje su korištene u industrijskom okruženju. Statističko testiranje temeljeno

na uporabi Musinih operacijskih profila, razvijeno u AT&T Bell Laboratories, uvriježena je tehnika za tu namjenu [15]. Ona je zasnovana na pretpostavci da će neispravnost unutar modula, koji sadrži češće izvođeni programski kod, vjerojatno utjecati na veći broj korisnika nego neispravnosti u ostalim modulima. S obzirom na prethodno navedeno, veći prioritet se dodjeljuje verifikaciji modula koji se češće izvode. Tvrtka AT&T je 1993. godine ovom tehnikom ostvarila omjer dobiti i troška veći od faktora 10 te umanjila trošak vremena i resursa u testiranju za 50% [16]. Ograničenje ove tehnike je sama procjena operacijskih profila jer se procjena programera i stvarno izvođenje programskog koda često ne podudaraju. Kritika ove tehnike jest njeno utemeljenje na suviše jednostavnom modelu koji uzima u obzir mali broj faktora [17]. Postoje i druge tehnike, koje teže napretku strategije verifikacije tako što usmjeravaju resurse verifikacije na češće korištene funkcije. Često su korištene tehnika temeljena na uporabi Markovljevih lanaca za izgradnju sekvencijskih modela programskog sustava [18] i tehnika temeljena na uporabi dijagrama stanja [19]. Sve navedene tehnike iziskuju vrlo zahtjevnu pripremu u vidu prikupljanja i analize podataka te izgradnje i simulacije modela za potporu u odlučivanju. Zajednički problem navedenih tehnika jest posebnost modela koji grade. Isti model nije moguće primijeniti više puta, što znači da se svaki put treba iznova uložiti veliki napor za pripremu [13, 16].

Modeli predviđanja kvalitete i programskih neispravnosti

U zahtjevnim procesima odlučivanja osiguranja kvalitete grade se analitički modeli procjene. Takvi modeli pružaju kvantitativnu procjenu kvalitete temeljem mjerenja određenih atributa programskog sustava. Njihova prednost je pružanje objektivnih mjerila trenutnog stanja kvalitete proizvoda, za razliku od subjektivnih procjena koje mogu biti pristrane osobi koja provodi procjenu, a samim time manje pouzdane i precizne [13]. Njihovom kontinuiranom primjenom moguće je čak i predviđati buduću kvalitetu, što uvelike može olakšati raspoređivanje poslova, resursa i rokova [13].

Podaci o neispravnostima sadrže informaciju koja ih vremenski pozicionira. Primjenom odgovarajućeg modela moguće je analizirati trend u njihovom ponašanju i predvidjeti njihovo ponašanje u budućnosti [13]. Vodeća empirijska istraživanja u programskom inženjerstvu usredotočuju se na određivanje modela distribucije neispravnosti unutar strukture velikih i složenih programskih sustava kroz njihovu evoluciju [3, 20]. Njihovi rezultati ukazuju da neispravnosti prate *Pareto* princip te se većina (oko 80%) neispravnosti nalazi u manjem dijelu (oko 20%) programskog koda [2–4]. Upravo takva distribucija motivira istraživanja u području predviđanja

programskih neispravnosti. Ono ima za cilj pronaći učinkovite modele predviđanja za lokaciju programskih modula koji su podložni neispravnostima. Takav model temelji se na mehanizmu prikupljanja podataka, predviđanja i donošenja preporuka koje bi pomogle stručnjacima za osiguranje kvalitete usmjeriti resurse u fazi validacije i verifikacije, te testiranja programskog sustava [21]. Boljim usmjeravanjem napora moguće je reducirati troškove i unaprijediti kvalitetu konačnog proizvoda. Značajan napredak u odnosu na postojeće tehnike predstavljala bi i mogućnost stvaranja opće važećih modela za različita okruženja primjene. Kako bi se istražila mogućnost izgradnje općenitih modela predviđanja neispravnosti, neophodno je provoditi istraživanja u što širem razvojnom okruženju. Potrebno je analizirati projekte iz različitih domena primjene, razvojnih zajednica ili faza zrelosti razvoja, s ciljem utvrđivanja utjecaja razvojnog okruženja na dobivene rezultate [20, 22, 23]. Nadalje, istraživači bi trebali pronaći način kako okarakterizirati okruženje i podatke koje koriste u svojim analizama [24].

1.2 Cilj istraživanja

Ono što se u začecima discipline nazivalo "*kriza programskih sustava*" (eng. software crisis) zbog sveobuhvatnog nedostatka javno objavljenih dobrih iskustava danas se nadoknađuje empirijskim istraživanjima [25]. Empirijska istraživanja uvelike su ovisna o kvaliteti podataka koje koriste, a podaci narušene kvalitete nisu dobra osnova za izgradnju novih teorija [26]. Poopćenje modela dobivenih iz studija slučaja otežano je upravo zato jer su rezultati najčešće pristrani nedovoljno precizno definiranim procesima koji se koriste već u prikupljanju podataka. Sukladno tome, sustavne smjernice za ispravno prikupljanje podataka postoje u raznim istraživačkim područjima. Međutim, u području predviđanja programskih neispravnosti evidentiran je nedostatak takvih smjernica. Trenutno stanje je takvo da postoje samo često korišteni postupci koji pak nisu bili kritički analizirani niti sustavno uspoređeni. Prvi cilj ove disertacije je napraviti sustavne smjernice za proces prikupljanja podataka za istraživanje u području predviđanja programskih neispravnosti koji nije pristran interpretacijama. Paralelno s predstavljanjem detalja problema, kojima se ova disertacija bavi, predstavljena su i aktualna znanstvena dostignuća evidentirana sustavnim pregledom literature. Analizom i usporedbom postojećih postupaka pronađen je presjek najboljih rješenja, a provedbom u stvarnom okruženju, dodatno je unaprijeđen novim saznanjima. Očekivani izvorni znanstveni doprinosi ove doktorske disertacije u kontekstu postupaka prikupljanja podataka su:

1. Postupak za prikupljanje podataka za potrebe predviđanja programskih neispravnosti s ciljem povećanja prikladnosti podataka;
2. Algoritam za prikupljanje podataka zasnovano na statičkim metrikama iz nestrukturiranih i formalno nepovezanih repozitorija programskog koda i neispravnosti.

Unapređenjem postupka za prikupljanje podataka i dobivanjem podataka veće točnosti, učinjen je samo prvi korak u uspješnoj provedbi predviđanja programskih neispravnosti. Daljnji predmet istraživanja u ovome području jest izgradnja odgovarajućeg modela za predviđanje. Dostupne su mnoge metode strojnog učenja za izgradnju takvog modela, ali stručnjacima i dalje nedostaju smjernice kako odabrati odgovarajuću metodu i kako procijeniti korist od njezine primjene. Budući da su mnoga istraživanja potvrdila kako opći najbolji model nije moguće pronaći neovisno o razvojnom okruženju, izgradnja smjernica trebala bi se orijentirati na opis razvojnog okruženja i uvjeta primjene unutar kojih će vrijediti određene pravilnosti. U mnogim aspektima programskog inženjerstva pokazalo se da je iluzorno očekivati pravilo koje će ukazati na jednu najbolju metodu za sve probleme. Umjesto toga, potrebno je pronaći parametre koji će pobliže opisati okruženje primjene i unutar kojeg će se ponuditi moguća rješenja za prevladavanje nađenih izazova [27]. S obzirom na prethodno rečeno, drugi cilj ove disertacije jest pronaći način na koji će se olakšati odabir odgovarajuće metode za izgradnju modela predviđanja programskih neispravnosti u stvarnoj primjeni programskog inženjerstva. Središte istraživanja je problem neujednačenosti skupova podataka temeljem kojih se model predviđanja treba izgraditi. Predložene su konkretne metode za utvrđivanje ovisnosti djelotvornosti modela predviđanja neispravnosti u odnosu na razinu neujednačenosti skupova podataka te su verificirane u primjeni nad stvarnim podacima. Očekivani izvorni znanstveni doprinosi ove doktorske disertacije u kontekstu odabira odgovarajuće metode strojnog učenja su:

3. Metoda za utvrđivanje granične razine neujednačenosti skupova podataka za izgradnju modela predviđanja programskih neispravnosti;
4. Postupak i verifikacija odabira kombinacije metoda za predviđanje programskih neispravnosti iz skupa metoda za pripremu podataka i metoda za strojno učenje.

Ova disertacija uvodi pojam **granična razina neujednačenosti**, kojim se podrazumijeva udio manjinske klase u skupu podataka kod kojeg nastupa najveća relativna promjena djelotvornosti modela predviđanja. Drugim riječima, to je svojevrsna točka prijeloma kod koje performanse odabrane metode strojnog učenja počinju naglo opadati.

1.3 Metodologija istraživanja

Cilj istraživanja ove disertacije usmjeren je na unapređenje postupaka koji služe za definiranje strategije verifikacije složenih programskih sustava. Empirijske metode su najpogodnije za postizanje tog cilja. *Studija slučaja* (eng. case study) je metodologija koja je općenito podobna za istraživanja u programskom inženjerstvu jer analizira suvremene fenomene u njihovom prirodnom okruženju [28]. Studije slučaja provode istraživanje u stvarnom industrijskom okruženju te koriste višestruke izvore dokaza, što dovodi do lakšeg planiranja, ali otežanog poopćenja rezultata [29]. Metodologije istraživanja koje su srodne studiji slučaja te se također često koriste su [28, 29]:

- *Pregled* (eng. survey) - sažima uobičajene informacije iz određene populacije ili njezina uzorka, najčešće putem anketa ili intervjua;
- *Eksperiment* (eng. experiment) - mjeri utjecaj na jednu varijablu manipulirajući drugom, uzimajući subjekte nasumično;
- *Aktivno istraživanje* (eng. action research) - metoda srodna studiji slučaja, koja za razliku od nje ne vrši samo promatranje, već je aktivno uključena u izmjene procesa kako bi analizirala njihov utjecaj.

Sustavan pregled literature

Sustavan pregled literature je sredstvo vrednovanja i interpretacije dostupnog istraživačkog korpusa koje se odnosi na neko istraživačko pitanje, područje ili fenomen. Svrha provođenja sustavnog pregleda literature može biti: sažimanje empirijskih spoznaja koje otkrivaju prednosti i nedostatke primjene neke metode, otkrivanje nedostataka u provedenim istraživanjima ili pozicioniranje vlastitog istraživanja u postojeće okvire [30]. Iako većina studija provodi pregled literature, njegova važnost je značajno manja ukoliko nije proveden u cijelosti i pravedno. Pravedno provođenje podrazumijeva uspostavu strategije pretraživanja koja će osigurati cjelovitost te nepristranost u izvještavanju studija koje podupiru ili ne podupiru hipoteze vlastitog istraživanja [30]. Prednost nad tradicionalnim pregledima literature je pružanje šire slike promatranog fenomena promatrajući ga sa svih strana, a moguća opasnost je da se zanemare manja odstupanja.

U domeni programskog inženjerstva postoje smjernice za provođenje sustavnog pregleda literature [30, 31]. One propisuju razinu detalja koju je nužno navesti kako bi se moglo analizirati, ponoviti, proširiti i kritički sagledati cjelovitost pregleda literature. Sustavan pregled

literature sastoji se od dva procesa: planiranja i provedbe. Faze unutar procesa planiranja su:

1. Utvrđivanje potrebe za sustavnim pregledom;
2. Definiranje protokola provedbe.

Faze unutar procesa provedbe sustavnog pregleda literature su:

1. Pronalazak studija;
2. Odabir primarnih studija;
3. Procjena kvalitete studija;
4. Prikupljanje korisnih podataka;
5. Sažimanje podataka.

Iako su faze navedene slijedno, njihovo izvođenje je često iterativno. Utvrđivanje potrebe postavlja cilj i istraživačko pitanje sustavnog pregleda te propituje ispravnost planiranog načina provedbe. Protokol provedbe definira izvore primarnih studija, ključne riječi za pretraživanje baza podataka, kriterije prihvaćanja i odbacivanja, kriterije za procjenu kvalitete, strategiju izvlačenja i sažimanja korisnih podataka te vremenskih rokova [30]. Protokol bi trebao omogućiti odgovaranje na pažljivo postavljena istraživačka pitanja koja će biti jasna u provedbi istraživanja i korisna u praktičnoj primjeni te za otkrivanje odstupanja među studijama. Prije nego se protokol počne primjenjivati, potrebno ga je usuglasiti među istraživačima koji ga provode.

Proces provedbe sustavnog pregleda literature prati i eventualno unapređuje prethodno definirani protokol. Pretražuju se baze podataka koje sadrže znanstveno istraživačke radove, relevantni časopisi i znanstveni članci, njihovi popisi literature i Internet općenito. Svaki korak se dokumentira radi ponovljivosti i provjere ispravnosti. Odabir studija temelji se na nedvosmislenim kriterijima prihvaćanja i odbacivanja u nekoliko faza: pretraga baza podataka, čitanje naslova, sažetka i cjelovitog rada. Svaka iduća faza podrazumijeva ulaganje većeg napora, ali nad sve manjim podskupom radova. Kvaliteta protokola se procjenjuje tako da se konzultira stručnjake u području koje pregled obuhvaća i testira na manjem podskupu. Kvaliteta prikupljenih radova procjenjuje se kroz potpunost metodologije pretraživanja, ispravnost parametara za odabir studija i druge čimbenike koji mogu narušiti značaj njihovih rezultata. Prikupljanje podataka temelji se na formama koje se definiraju u protokolu, a mogu se unaprijediti prilikom provedbe. Poželjno je da podaci budu izraženi kvantitativno radi lakše provedbe naknadnih analiza te uspoređeni od strane više istraživača koji provode pregled. Sažimanje podataka može uključiti opisnu analizu koja tablično predstavlja prikupljene podatke, njihove sličnosti i njihove različitosti. Kvantitativna analiza ovisi o tipu podataka, usporedivosti procesa i postupaka

korištenih u prikupljenim studijama te njihove kvalitete [30].

Analiza slučaja

Analiza slučaja je fleksibilna metoda, što je neophodno u istraživanju složenih i dinamičkih svojstava fenomena koje je teško razgraničiti od njezina okruženja [32]. Upravo takvi složeni fenomeni istražuju se u programskom inženjerstvu jer se promatraju pojedinci, grupe i organizacije u provođenju aktivnosti razvoja, dizajna, implementacije i održavanja programskih sustava u različitim uvjetima [28]. Analiza slučaja je stoga najčešće korištena metoda istraživanja u programskom inženjerstvu. Bez obzira na prethodnu podjelu, analiza slučaja često sadrži elemente pripisane drugim metodama. Primjerice, pregled literature je primjeren uvod u analizu slučaja, a podaci se prikupljaju iz pregleda arhiva ili anketa i intervjua. Za razliku od eksperimenata i pregleda, dizajn procesa analize slučaja također je fleksibilan jer se definiraju samo temeljni parametri, a ostali se mogu naknadno prilagoditi. Iako podaci mogu biti kvantitativni ili kvalitativni, teži se uporabi kvantitativnih podataka radi postizanja bogatijih opisa promatranog fenomena [32]. Svrha analize slučaja može biti ostvarivanje različitih zadaća, a općenito ih se svrstava u četiri kategorije [29]:

- *Istraživanje* (eng. exploration) - trenutnog stanja ili novih spoznaja u promatranom fenomenu koje potiče stvaranje novih ideja i hipoteza;
- *Opis* (eng. description) - situacije promatranog fenomena;
- *Objašnjenje* (eng. explanation) - temeljeno na zavisnim odnosima koje traži uzrok promatranog fenomena;
- *Unapređenje* (eng. improvement) - određenog aspekta promatranog fenomena.

Analize slučaja u programskom inženjerstvu prvotno su bile namijenjene istraživanju. Ukoliko je poopćenje promatranog fenomena od sekundarne važnosti, mogu biti namijenjene opisu. Koriste se i za objašnjenje iako odvajanje utjecaja međuovisnih čimbenika na promatrani fenomen može biti problematično. Konačno, mogu se uporabiti i za unapređenje brojnih procesa unutar discipline programskog inženjerstva [28]. U konačnici, analiza slučaja trebala bi istražiti statistički značaj dobivenih rezultata prije nego li donese zaključke. Pri tome je uputno priložiti i ostale dokaze, slike, izjave, dokumente i slične međukorake koji podržavaju zaključke i pridodaju njihovom značaju [28]. Za postizanje veće preciznosti empirijskih studija, predlaže se uporaba triangulacije. Ona podrazumijeva sagledavanje istog fenomena iz različitih perspektiva kako bi se dobilo širu sliku. Moguće je uporabiti triangulaciju podataka koristeći više različi-

tih izvora, triangulaciju promatrača, odnosno osoba koje prikupljaju podatke, te triangulaciju metoda i teorija kako bi se pokazala nepristranost rezultata.

Koraci koje je važno provesti u analizi slučaja moraju se osvrnuti na pet velikih procesa:

1. Dizajn analize slučaja koji uspostavlja cilj i plan istraživanja;
2. Pripremu procedura i protokola za prikupljanje podatka;
3. Provedbu prikupljanja podataka i dokaza;
4. Analizu prikupljenih podataka;
5. Izvještaj dobivenih rezultata.

Valjanost istraživanja

U analizi slučaja važno je razmotriti čimbenike koji su mogli narušiti valjanost i ispravnost rezultata istraživanja. Pritom se razlikuje: *valjanost konstrukcije* (eng. construct validity), *valjanost zaključaka* (eng. conclusion validity), *unutarnja valjanost* (eng. internal validity) i *vanjska valjanost* (eng. external validity) [28]. Valjanost konstrukcije propituje jesu li uporabljene metode mjerile ono za što su namijenjene. Provedba istraživanja u industrijskom okruženju stvarnih projekata osnažuje valjanost konstrukcije. Valjanost zaključaka, odnosno njihova pouzdanost propituje je li moguće donijeti zaključke iz dobivenih rezultata. To se ponajprije veže za ponovljivost mjerenja, odnosno jesu li dobiveni podaci pristrani osobama koje su ih prikupljale ili istraživaču koji ih analizira [29]. Unutarnja valjanost, odnosno kauzalnost provjerava omogućuje li dizajn studije izvođenje zaključaka iz uzroka i ishoda. Najveća poteškoća može pri tome biti utjecaj čimbenika kojih istraživač nije niti svjestan. Vanjska valjanost propituje mogućnost *poopćenja* (eng. generalizability) rezultata. Ovo je velik problem u analizama slučaja, budući da se one odnose na konkretan slučaj i okruženje koje ga definira [29].

1.4 Pregled rada

U pozadini istraživanja provedenog ovom disertacijom nalazi se potreba uspostave detaljnih smjernica koje bi pomogle u provedbi uspješne i usporedive primjene predviđanja programskih neispravnosti. Važnost kvalitete podataka je pri tome od velikog značaja jer o njima ovisi valjanost zaključaka dobivenih njihovom analizom. Utvrđivanjem postupka i algoritma za prikupljanje podataka kojim bi se postigla najveća prikladnost osigurala bi se zajednička osnova za usporedivost zaključaka budućih istraživanja. Utvrđivanjem metode za pronalazak granične

razine neujednačenosti podataka stručnjaci za kvalitetu bi imali jasne naputke, na koji način analizirati i vrednovati vlastite podatke, te smjernice kako odabrati model predviđanja prikladan njihovom slučaju. Upravo su to i temelji istraživačkih pitanja koja vode ovu disertaciju. Najkraće rečeno, ona su:

(1) Kako provesti prikupljanje podataka?

(2) Kako izgraditi uspješan model predviđanja?

Metodologija koja se koristi za dobivanje odgovora na postavljena pitanja uključuje:

- Sustavan pregled literature;
- Analizu parametara u postojećim pristupima prikupljanju podataka;
- Nedvosmislenu i jasnu definiciju pristupa prikupljanju podataka;
- Opsežnu komparativnu studiju utvrđivanja najboljeg pristupa prikupljanju podataka;
- Prijedlog metode za utvrđivanje utjecaja razine neujednačenosti skupova podataka;
- Empirijsko istraživanje utjecaja razina neujednačenosti skupova podataka.

Sadržaj poglavlja ove disertacije prati provedbu predložene metodologije. Poglavlje 2 bavi se detaljima koji leže u pozadini problema s kojima se ova disertacija bavi. Opisani su mogući izvori podataka za predviđanje programskih neispravnosti, popularno korištene procedure njihovog prikupljanja i tehnike povezivanja formalno nepovezanih repozitorija. Opisan je i proces provedbe predviđanja programskih neispravnosti, predstavljen je problem neujednačenosti skupova podataka te opis mehanizama metoda strojnog učenja odabranih za empirijsko istraživanje.

Poglavlje 3 ulazi dublje u probleme prikupljanja podataka za predviđanje programskih neispravnosti. Predstavljen je sustavan pregled literature čiji je cilj bio otkriti i diskutirati parametre prikupljanja podataka u postojećim pristupima. Njihova precizna definicija je važna jer može rezultirati odstupanjem u prikupljenim podacima ukoliko ih se ostavi otvorenima za različite interpretacije. Pod pojmom **parametri prikupljanja podataka** podrazumijevaju se svi parametri unutar procesa prikupljanja podataka, koji mogu imati više različitih vrijednosti, i zahtjevi koje treba ispuniti. To se odnosi na parametre poput razine težine neispravnosti, redoslijeda prikupljanja podataka ili odluke što učiniti s dvostrukim podacima. To se također odnosi na zahtjeve poput karakteristika alata za izračun metrika programskog koda ili projekata iz kojih je uopće moguće provesti prikupljanje. U nastavku poglavlja predstavljena je i istraživačka studija. Njezin cilj bio je kvantificirati moguća odstupanja u prikupljenim podacima te pronaći najprikladnije alate za izračun metrika programskog koda. Ovo poglavlje je područje rada [33].

Poglavlje 4 nastavlja se na rezultate dobivene iz prethodnog poglavlja. U njemu je predstavljen alat koji implementira sustavno definirani postupak prikupljanja podataka i novu tehniku povezivanja razvojnih repozitorija. Radi utvrđivanja ispravnosti razvijenog algoritma provedena je i prikazana analiza slučaja koja daje dobre rezultate. Ovo poglavlje je područje radova [34] i [35].

Poglavlje 5 završava istraživanje vezano za prikupljanje podataka. U njemu je predstavljen okvir za provedbu detaljne komparativne studije koja ima za cilj utvrditi najuspješniji pristup prikupljanju podataka za predviđanje programskih neispravnosti. Potreba za ovakvom studijom proizašla je iz nedostatka jasno propisanih procesa kojima se prikupljanje provodi. Značaj ove studije je uspostava okvira za vrednovanje i za buduće tehnike povezivanja razvojnih repozitorija ili pristupa prikupljanju. Rezultati ovog poglavlja ukazuju da predloženi postupak prikupljanja i razvijeni algoritam za povezivanje formalno nepovezanih repozitorija daju podatke najveće točnosti. Ovo poglavlje je područje rada objavljenog u međunarodnom časopisu [36].

Poglavlje 6 osvrće se na problem neujednačenosti skupova podataka koji se koriste za opis okruženja primjene metoda strojnog učenja. U poglavlju se predstavlja metoda za utvrđivanje granične razine neujednačenosti od koje metode strojnog učenja počinju davati lošije rezultate, te metoda za utvrđivanje najuspješnije metode strojnog učenja s obzirom na prisutnu razinu neujednačenosti u skupovima podataka. Provedbom empirijskog istraživanja pokazano je da predložene metode uspješno obavljaju svoj zadatak. Ovo poglavlje je nastavak istraživanja započetog u radovima [37] i [38].

Poglavlje 7 sadrži dva priloga. Prvi prilog objašnjava objektno orijentirane metrike programskog sustava koje se prikupljaju razvijenim alatom. Te metrike ujedno predstavljaju nezavisne varijable na kojima se temelji predviđanje programskih neispravnosti koje je provedeno u ovoj disertaciji. Drugi prilog sadrži sve detalje rezultata statističkih analiza i testova empirijskog istraživanja iz poglavlja 6.

Poglavlje 2

Definicija problema i metode strojnog učenja

Brojna istraživanja tražila su model za predviđanje programskih neispravnosti koji će imati najbolje performanse [39]. Postoji velik broj metoda strojnog učenja, ali nije pronađena općenito najbolja. Stručnjacima i dalje nedostaju upute o tome koju bi metodu valjalo primijeniti u određenim uvjetima. Pored toga, rezultati tih istraživanja teško su usporedivi zbog inherentnih svojstava metoda predviđanja i postupaka kojima se istraživanja provode [40–42]. Otkriveno je nekoliko uzroka odstupanja u postupcima istraživanja koji doprinose ovom problemu, poput odabira postupaka za predobradu i uzorkovanje podataka, pristranosti samog istraživača ili pristranosti odabrane mjere za vrednovanje modela predviđanja [43]. Ograničena uporaba statističkih procedura u testiranju modela predviđanja i analizi značajnosti dobivenih rezultata također predstavlja prijetnju valjanosti provedenih istraživanja [44]. Iz svega navedenog proizlazi potreba za sustavno definiranim smjernicama za provedbu istraživanja, od prikupljanja podataka do analize rezultata, kako bi se smanjio utjecaj prijetnji valjanosti istraživanja. Dakako, sam odabir podataka, kao i njihove posebnosti, također mogu utjecati na ponašanje modela za predviđanje [42]. Stoga, za usporedbu različitih modela predviđanja potrebno je pronaći uvjete u kojima oni imaju konzistentno ponašanje.

Velik problem u području predviđanja programskih neispravnosti jest nepoznavanje uzroka pristranosti rezultata, zbog čega nije moguće pronaći općenito važeće zaključke [39]. Istraživanja u ovome području oslanjaju se na podatke dobivene iz repozitorija namijenjenih fazi razvoja programskih sustava. To su repozitoriji sustava za praćenje zahtjeva i sustava za upravljanje inačicama koji nisu formalno povezani [45]. Uporabom različitih tehnika, moguće je

povezati strukturirane i nestrukturirane informacije iz tih repozitorija. Uspješno povezivanje tih podataka predstavlja izvor informacija esencijalan za otkrivanje ponašajnih modela složenih struktura programskih sustava. ***Povezivanje podataka*** (eng. data linking) prepoznato je kao snažan alat za pronalazak i razvoj novih teorija i u drugim istraživačkim granama, poput zdravstva [46], nacionalne statistike [47], bioinformatike, klimatskih promjena i sl. Brojna istraživanja u domeni programskog inženjerstva razvijala su tehnike i alate za povezivanje razvojnih repozitorija, kao što su sljedivost zahtjeva [48], povezivanja neispravnosti i koda [49], te lokalizacija neispravnosti [50, 51]. Međutim, postojeće tehnike nisu dostigle zadovoljavajuću razinu točnosti, a njihova međusobna odstupanja navode se kao velik problem [52]. Također, do sada nije provedena sustavna usporedba postojećih tehnika i uobičajeno su korištene bez da su se detaljno analizirale. Iz svega navedenog slijedi potreba za izgradnjom sustavno definiranog procesa prikupljanja podataka iz razvojnih repozitorija [53]. Taj je problem riješen ovom disertacijom. Istraženi su svi parametri u postojećim pristupima prikupljanja podataka te su za njih definirani nedvosmisleni kriteriji. Provedena je analiza slučaja nad testnim skupom podataka kako bi se ustanovila valjanost postojećih tehnika za povezivanje razvojnih repozitorija. Razvijena je nova tehnika za povezivanje repozitorija koja nadilazi slabosti postojećih. Definirani parametri prikupljanja podataka i nova tehnika povezivanja repozitorija sačinjavaju sustavno definiran proces prikupljanja podataka. Potom je provedena opsežna komparativna studija sustavno definiranog procesa prikupljanja podataka s postojećim tehnikama i procesima. Na temelju dobivenih rezultata, razvijen je alat za automatiziranu provedbu prikupljanja podataka koji je uporabljiv u široj istraživačkoj zajednici i industrijskoj primjeni.

Zajednički problem svim metodama predviđanja je **neujednačenost skupova podataka**. Poznato je da ovaj problem može utjecati na odabir algoritma za klasifikaciju [54] i tehnike predobrade podataka [55, 56]. Budući da je taj problem inherentan za predviđanje programskih neispravnosti, u ovoj disertaciji uporabljen je kao zajednički uvjet koji može usmjeriti izgradnju modela predviđanja. Provedena je opsežna empirijska studija koja je analizirala ponašanje odabranih metoda predviđanja u uvjetima različitih razina neujednačenosti skupova podataka. Predložena je metoda za analizu utjecaja razine neujednačenosti skupova podataka u odabranom okruženju primjene, kako bi se moglo odrediti odgovarajuću metodu strojnog učenja za izgradnju modela predviđanja. Iako je istraživanje provedeno u domeni predviđanja programskih neispravnosti, metoda je primjenjiva i u drugim istraživačkim područjima koja koriste metode strojnog učenja u okruženju neujednačenih skupova podataka.

2.1 Podaci za predviđanje programskih neispravnosti

Predviđanje programskih neispravnosti temelji se na modelu naučenom na povijesnim podacima. Uspješnost modela ovisi o kvaliteti postupaka kojima se prikupljaju podaci. S obzirom na izvor informacija, metode prikupljanja podataka dijelimo na direktne (npr. intervju), indirektne (npr. korištenje mjernih instrumenata) i neovisne (npr. pregled dokumentacije) [2]. Prikupljanje za SDP može se opisati kao kombinacija indirektnog i neovisnog jer se koriste alati za mjerenje metrika programskog koda i pregledava se dokumentacija o neispravnostima. Metodologija ispravnog prikupljanja podataka u programskom inženjerstvu definirana je još davne 1984. godine [57]. Iako je prvotno namijenjena direktnom prikupljanju podataka, neke su smjernice opće važeće. Procedura prikupljanja podataka mora biti provjerljiva, ponovljiva i precizna [57]. Potrebno je ustanoviti cilj prikupljanja kako bismo definirali cjelovit popis relevantnih informacija, odredili kategorije podataka i pripremili formu za prikupljanje. U kontekstu predviđanja programskih neispravnosti, navedeni koraci nisu naročito problematični. Međutim, važno je i osigurati provjeru valjanosti kako bi se moglo kvantificirati točnost prikupljenih podataka. Taj je korak vrlo često zanemaren u domeni predviđanja programskih neispravnosti, a mnoge studije čak koriste podatke za koje ne postoji opis procesa prikupljanja [39].

Izvori podataka u istraživačkim radovima ovoga područja mogu se svrstati u tri kategorije:

1. Repozitoriji gotovih skupova podataka;
2. Podaci iz industrijskih projekata;
3. Razvojni repozitoriji projekata otvorenog koda.

Gotovi skupovi podataka predstavljaju veliki iskorak jer su olakšali pristup podacima široj istraživačkoj zajednici i omogućili provedbu studija na zajedničkim osnovama. Međutim, najpopularniji repozitoriji podataka našli su se i na udaru kritika zbog uočenih propusta u kvaliteti podataka, ali i izostanku procedure i opisa okruženja njihovog prikupljanja [39, 58, 59]. Podaci iz industrijskih projekata također moraju imati jasno definiranu proceduru prikupljanja te svakako predstavljaju najvredniji izvor informacija. Problem industrijskih podataka je što gotovo nikada nisu dostupni široj zajednici zbog tajnosti ili osjetljivosti informacija koje sadrže. Zbog svega navedenog, sve veća važnost pridaje se prikupljanju podataka iz razvojnih repozitorija projekata otvorenog koda. Osim dostupnosti prijeko potrebnih podataka, oni omogućavaju provedbu, usporedbu i unapređenje procedura prikupljanja. U vodećim razvojnim zajednicama kao što su Eclipse, Apache ili Mozilla mogu se pronaći i projekti čija su veličina, složenost i dugovječnost usporedivi s industrijskim projektima.

Sustav za upravljanje inačicama

Razvoj modernih programskih sustava koji postaju sve veći i složeniji, nezamisliv je bez upotrebe *sustava za upravljanje inačicama* (eng. version control system). Sustav za upravljanje inačicama služi za pohranu izmjena programskog koda i omogućuje paralelan rad većeg broja programera. On predstavlja najvredniji izvor podataka u analizama slučaja koje koriste arhivske podatke, naročito u studijama osiguranja kvalitete [32]. Budući da svrha takvih sustava nije pružanje informacija za provedbu istraživanja, važno je ispravno izdvojiti korisne informacije [32]. Takvi sustavi koriste distribuiranu pohranu podataka te omogućuju pohranu cjelokupne povijesti izmjena programskog koda uz minimalno opterećenje memorije, pristup prošlim i stabilnim inačicama koda, istovremeni razvoj više inačica nekog programskog sustava i pristup kodu u globaliziranom razvoju. Više programera može raditi na istom projektu tako što svatko lokalno stvara svoju radnu kopiju projekta te prilaže *predaju* (eng. commit) vlastitih izmjena. Predaja je atomarna operacija koja sustavu nalaže da određenu grupu izmjena učini konačnima za sve korisnike. Predaja je uspješna ukoliko nije u konfliktu s predajama koje su priložene nakon što je stvorena lokalna radna kopija. Starije inačice su se osiguravale od konflikata tako da bi zaključale datoteku koju neki programer odluči mijenjati. Novi sustavi omogućuju *spajanje inačica* (eng. version merging), kod kojeg prva izmjena uvijek prolazi, a svaka iduća izmjena na istoj datoteci mora biti usklađena s prethodnom. Tada se pohranjuju:

- Sve datoteke koje su izmijenjene na radnoj kopiji;
- Identifikacijska oznaka - ID predaje;
- Datum predaje;
- Ime programera;
- Opis izmjene koju upisuje programer;
- Temeljne oznake, etikete i oznake.

Posljednje navedene služe kao dodatne oznake za definiranje faza razvoja projekata poput *izdanja* (eng. release), *grananja* (eng. branch) i *miljokaza* (eng. milestone). Sustavi često izračunaju i količinu izmjena izraženu brojem izbrisanih, izmijenjenih i dodanih linija koda. Budući da takvi sustavi sadrže cjelokupni dnevnik izmjena, odnosno evolucije projekata, njihovi repozitoriji predstavljaju vrlo važan izvor informacija za razvojni proces [60]. Sustavi poput Subversion (SVN)* ili Git† često su korišteni u razvoju projekata otvorenog i zatvorenog koda. Najpopular-

*http://subversion.tigris.org/
†https://git-scm.com/

niji servisi[‡] koji pružaju usluge tih sustava su GitHub (preko 8 milijuna korisnika i 19 milijuna projekata), BitBucket (preko 2,5 milijuna korisnika i 330 tisuća projekata) i SourceForge (preko 3 milijuna korisnika i 320 tisuća projekata).

Sustav za praćenje zahtjeva

Pored sustava za upravljanje inačicama, moderni projekti ne mogu se razvijati niti bez *sustava za praćenje zahtjeva* (eng. issue tracking system). Zahtjeve možemo pri tome podijeliti na *unapređenja* (eng. enhancement) i *neispravnosti* (eng. defect). Neispravnosti su neizbježne prilikom izrade programskog sustava, a njihovo sustavno praćenje vrlo je važno za uspješno upravljanje kvalitetom [60]. Sustave za praćenje zahtjeva često nazivamo i *sustavima za praćenje programskih neispravnosti* (eng. bug tracking system) jer pružaju repozitorij prilagođen za praćenje cjelokupnog životnog ciklusa programske neispravnosti. Klasični životni ciklus, odnosno proces ispravljanja neispravnosti uključuje pronalazak i dijagnosticiranje problema, dizajniranje i pisanje rješenja, te integraciju ispravljenog modula s ostatkom sustava [61]. Moderni sustavi omogućuju praćenje navedenih faza kroz predefinirane kategorije atributa statusa neispravnosti. Preostali atributi koji se pri tome dodjeljuju zahtjevima služe za njihovu jedinstvenu identifikaciju, te opisuju programski sustav za koji se zadaju, okolnosti u kojima su nastali, njihovu važnost, postupak razrješavanja i sl. Norma IEEE 1044-2009 propisuje listu atributa koju bi organizacije trebale navoditi prilikom praćenja oštećenja, odnosno neispravnosti [10]:

- ID neispravnosti - jedinstvena identifikacijska oznaka (ID) neispravnosti;
- Opis - sažetak o tome što je pogrešno, nepotrebno ili nedostaje;
- Status - definira trenutnu fazu u životnom ciklusu neispravnosti;
- *Resurs* (eng. asset) - programska struktura koji sadrži neispravnost;
- Artefakt - zaseban dio programskog sustava koji sadrži neispravnost;
- Inačica detekcije - inačica programskog sustava u kojoj je neispravnost otkrivena;
- Inačica korekcije - inačica programskog sustava u kojoj je neispravnost ispravljena;
- Prioritet - rangiranje važnosti rješavanja neispravnosti u odnosu na ostale;
- Težina - najozbiljnija posljedica koju može uzrokovati neispravnost;
- Vjerojatnost - vjerojatnost ponovne pojave kvara uzrokovanog ovom neispravnošću;
- Utjecaj - klasa zahtjeva na koje utječe kvar uzrokovan ovom neispravnošću;

[‡]http://software.ac.uk

- Tip - opis temeljen na razredu programskog koda ili proizvoda unutar kojeg se neispravnost pronalazi;
- Način - opis manifestacije neispravnosti koji govori je li nešto suvišno ili je nešto izostavljeno u implementaciji ili reprezentaciji;
- Ponovljivost - aktivnost tijekom koje se neispravnost manifestira;
- Detekcija - aktivnost tijekom koje je neispravnost uočena (npr. inspekcija ili testiranje);
- Referentni kvar - sredstvo otkrivanja kvara uzrokovanog ovom neispravnošću;
- Referentna izmjena - sredstvo otkrivanja zahtjeva za uklanjanje neispravnosti;
- Rezolucija - konačni status neispravnosti prilikom zatvaranja (ispravljena, dvostruko prijavljena, nemoguće ponoviti, nemoguće ispraviti).

Nekomercijalni i najčešće korišteni sustav za praćenje neispravnosti je Bugzilla[§]. Bugzilla i slični sustavi omogućuju praćenje cjelokupnog životnog ciklusa neispravnosti i zahtjeva za unapređenje. Atribut statusa neispravnosti prepoznaje sljedeće kategorije neispravnosti: nova, verificirana, dodijeljena i zatvorena. Upravo atributi statusa i rezolucije služe za praćenje faza u kojima se nalazi neka neispravnost. Model težine neispravnosti razlikuje sljedeće kategorije: blokirajuće, kritične, velike, normalne, malene, trivijalne, te unapređenje. Tim atributom može se okarakterizirati težina neispravnosti, ali se može i razlikovati neispravnosti od zahtjeva za unapređenje. Pored navedenih atributa koje definira norma IEEE 1044-2009, ovi sustavi nude i neke dodatne atribute koji omogućuju diskusiju o problemu i praćenje napretka ispravljanja:

- Ime i/ili e-adresa osobe koja prijavljuje neispravnost - prijavitelj;
- Ime i/ili e-adresa osobe kojoj je neispravnost dodijeljena - zadužena osoba;
- Datum prijave i posljednje izmjene statusa;
- Prilozi (npr. preslika ekrana);
- Komentari koji prate diskusiju.

Sve nadvedene informacije predstavljaju vrijedan izvor za analizu raznih faza procesa programskog inženjerstva. Projekti otvorenog koda koji koriste takve sustave omogućuju javan pristup njihovim repozitorijima. Štoviše, imaju otvoren pristup i za prijave neispravnosti pa i kvaliteta tih podataka može biti narušena neželjenim i višestrukim unosima ili neispravnom kvalifikacijom [62]. Problem ponekad uzrokuje i činjenica da korisnici i programeri različito shvaćaju neispravnosti i zahtjeve za unapređenje [63]. Ove nedostatke donekle kompenzira hijerarhijska struktura razvojnih zajednica projekata otvorenog, koda koja ipak uvodi sustavnost

[§]http://www.bugzilla.org/

i nadzor podataka od strane najiskusnijih članova. Najčešće je korišten *model koncentričnih krugova* (eng. onion model) koji podrazumijeva sljedeće krugove, poredane počevši od najvažnijeg: *temeljni programeri* (eng. core developers), aktivni programeri, periferni programeri, *programeri zakrpa* (eng. patch submitters), aktivni korisnici i pasivni korisnici [64]. Oni najvažniji krugovi imaju najveću težinu u procjeni podataka koji su uneseni te mogu unaprijediti njihovu kvalitetu.

Analiza neispravnosti često svrstava neispravnosti u dvije kategorije, one koje su otkrivene prije isporuke i one koje su otkrivene poslije isporuke [45, 65]. Neispravnosti otkrivene prije isporuke su one koje se otkriju prilikom razvoja i testiranja programskog sustava, dok su neispravnosti otkrivene nakon isporuke one koje se pojave prilikom uporabe programskog sustava, odnosno one koje utječu na korisnika [65]. Usporedbom datuma prijave neispravnosti iz sustava za praćenje zahtjeva i datuma isporuke, neispravnosti se svrstaju u odgovarajuću kategoriju. Analize neispravnosti koje razlikuju neispravnosti otkrivene prije i poslije isporuke mogu biti korisne jer cijena neispravnosti nije jednaka za ove dvije kategorije.

2.1.1 Provedba prikupljanja podataka

Prikupljanje podataka za SDP je složen zadatak jer podrazumijeva preuzimanje i povezivanje podataka koji su spremljeni u formalno nepovezanim razvojnim repozitorijima [45]. Problem predstavlja i nedostatak normi u definiciji metrika programskog sustava. Za provedbu valjane analize slučaja, nužno je koristiti višestruke izvore podataka što dodatno otežava stvari ukoliko se pritom ne koriste tehnike s mogućnošću automatizirane provedbe prikupljanja podataka [32]. Uobičajeni skup podataka sastoji se od nezavisnih varijabli koje opisuju module programskog sustava te od broja neispravnosti kao zavisne varijable. No, kada se takav skup podataka želi prikupiti, počinju se pojavljivati mnoga pitanja. Potrebno je definirati nezavisne varijable, koje mogu biti razni parametri veličine i složenosti programskog koda, kao i razinu granulacije na kojoj se programski sustav promatra, te vrstu neispravnosti koje tražimo. Također treba odabrati alate i odgovarajuću tehniku za povezivanje neispravnosti i programskog koda. Sva ta pitanja objedinjujemo pod nazivom procedura prikupljanja podataka. Tehnika povezivanja jedan je od dijelova procedure prikupljanja podataka koji je obrađen u nastavku ovog poglavlja. Iako je to najvažniji dio procedure, i ostala pitanja moraju imati jasne kriterije jer mogu dovesti do velikih razlika u skupovima podataka.

Razlikujemo dva moguća pristupa u provedbi povezivanja razvojnih repozitorija. Prvi je

deterministički pristup. On se nameće kao najbolje rješenje, jer podrazumijeva pronalazak jedinstvenih oznaka koje bi trebale biti prisutne u oba repozitorija [66]. Takav pristup može poslužiti kao dobra tehnika povezivanja ukoliko je ključ povezivanja precizan, stabilan i dostupan. Drugi pristup, probabilistički, traži skup zajedničkih atributa unutar dvaju repozitorija. Veza je uspostavljena ukoliko se većina atributa podudara, odnosno nije uspostavljena ukoliko se atributi pretežno razilaze [67].

Popularni **pristup SZZ** (Sliwerski, Zimmermann, Zeller [68]) primjer je kombinacije determinističkog i probabilističkog pristupa. Deterministički se traže ID neispravnosti i neke ključne riječi unutar opisa predaja. Potom se svakoj potencijalnoj vezi neispravnosti i predaja dodjeljuje sintaktička i semantička težina [68]. Sintaktička težina inkrementira se ako unutar opisa predaje sadrži ID neispravnosti, ako uporabom regularnih izraza pronađe riječi poput "*fix*", "*bug*" "*defect*", "*patch*" ili znak "#" koji je postavljen kao prefiks ID-u ili ako pored ID-a ne sadrži nikakvih riječi. Semantička težina inkrementira se ako se ID referencira na ispravljenu neispravnost, ako opis predaje sadrži opis neispravnosti, ako se ijedna izmijenjena datoteka spominje unutar opisa neispravnosti ili ako je osoba zadužena za neispravnost ujedno i autor izmjene programskog koda (kasnije definirano kao tehnika podudaranje autorstva). Potom je empirijski ustanovljena granična vrijednost sintaktičke i semantičke težine po kojoj se probabilistički prihvaćaju ili odbacuju moguće veze. Prihvaćene veze moraju imati semantičku težinu veću od 1 ili semantičku i sintaktičku težinu barem 1. No, uporaba višestrukih ključeva povezivanja može narušiti kvalitetu podataka, kao što je to uočeno u domeni zdravstva [66]. Pristup SZZ je najbolje opisani proces i najčešće je korišten u srodnoj literaturi. To dokazuje velik broj studija koje ga primjenjuju ([69–71, 71–77], itd.) ili koriste javno dostupne podatke za tri uzastopne inačice sustava Eclipse koji su dobiveni primjenom pristupa SZZ ([78–82].

Tehnike povezivanja razvojnih repozitorija

Odabir odgovarajuće tehnike povezivanja formalno nepovezanih razvojnih repozitorija jedan je od sastavnih dijelova procedure prikupljanja podataka. Tim se problemom posvetio i određen broj studija. Ustanovljeno je da kvaliteta povezanih podataka ovisi o odabiru tehnike povezivanja repozitorija [83]. Izostanak normi često dovodi do odstupanja, a ona onemogućuju poopćenje rezultata. Međutim, kako tehnike međusobno nisu uspoređivane, nije poznato u kojoj mjeri mogu utjecati na pristranost podataka te umanjiti značaj rezultata istraživanja koja ih koriste. *Odstupanje u povezivanju* (eng. linking bias) može dovesti do pogrešno uspostavljene veze ili

do izostanka uspostave veze. No, čak i uz idealnu tehniku povezivanja, neke neispravnosti ne mogu se povezati zbog ljudske pogreške prilikom unosa podataka u repozitorije [84]. Postotak neispravnosti koji se poveže s barem jednom predajom nazivamo *stopa povezivanja* (eng. linking rate). Dodatni problem za kvalitetu podataka mogu predstavljati i neispravnosti koje nisu unesene u repozitorij sustava za praćenje zahtjeva. Takve neispravnosti ponekad znaju biti samo spomenute u korespondenciji programera [85]. Međutim, taj je problem izvan područja istraživanja ove disertacije. Postoji nekoliko tehnika koje se zasnivaju na povezivanju strukturiranih podataka ili nestrukturiranih podataka. Sljedeći prijevodi s engleskog jezika koriste se kao imena tehnika povezivanja u ostatku ove disertacije:

1. *Jednostavna pretraga* (eng. simple search) identifikacijske oznake neispravnosti;
2. *Pretraga regularnim izrazima* (eng. regular expression search) za ključnim terminima;
3. *Podudaranje autorstva* (eng. authorship correspondence);
4. *Vremenska korelacija* (eng. time correlation) aktivnosti u fazi ispravljanja neispravnosti;
5. *Obrada prirodnog jezika* (eng. natural language processing) nestrukturiranih informacija.

Jednostavna pretraga traži broj *identifikacijske oznake* (ID) *neispravnosti* (eng. Bug ID) unutar opisa predaje izmjene programskog koda [35]. Međutim, u repozitorijima većine sustava za upravljanje inačicama nije obavezno navoditi ID neispravnosti iako se izmjenom koda ona ispravlja. Posljedično, jednostavna pretraga oslanja se na navike programera i običaje razvojnih zajednica te ne uspijeva pronaći sve poveznice. Ustanovljena je pozitivna korelacija između iskustva programera kod dugotrajnih projekata i uspješnosti povezivanja [86].

Kako bi se unaprijedilo točnost uspostave veze između neispravnosti i predaja, pretragu se proširilo na druge informacije unutar repozitorija. **Pretraga regularnim izrazima** (Regex) je tehnika koja traži ključne riječi unutar opisa predaja. Regularnim izrazom se formira obrazac pretrage kojim se traži podudaranje unutar znakovnih nizova za *pretraživanje informacija* (eng. information retreival) [87]. Obrazac pretrage može sadržavati regularne i *meta-znakove* (eng. meta-characters), koji se zajednički koriste za davanje konciznog opisa skupa znakova. Prva uporaba tehnike Regex tražila je podudaranje imena datoteka koje su mijenjane u repozitoriju sustava za upravljanje inačicama i spominjane unutar repozitorija sustava za praćenje neispravnosti [88].

Podudaranje autorstva je tehnika koja povezuje imena autora predaje izmjena programskog koda s imenom osobe zadužene za rješavanje iste neispravnosti [49]. Ona polazi od pretpostavke da su stvarno ime, korisničko ime ili nadimak programera identično navedeni u oba

repozitorija te da je svaku ispravljenu neispravnost riješila upravo ona osoba koja je za nju bila nadležna. Dok se prva pretpostavka najčešće ispuni, druga ponekad nije ispunjena jer podjela zadataka u zajednicama otvorenog koda nije toliko stroga. Naročito kod složenih sustava, kod kojih je funkcionalnost različitih modula zavisna, nije neuobičajeno da programer riješi tuđe neispravnosti prilikom ispravljanja svoje [85].

Tehnika **vremenske korelacije** povezuje vremena izmjene statusa neispravnosti i vremena predaje izmjene programskog koda. Jedna inačica ove tehnike traži potvrdu ispravnosti veza koje su prethodno detektirane tehnikom jednostavne pretrage ID-a ili pretragom ključnih riječi tehnikom Regex [60]. Veza bi bila potvrđena ukoliko je status neispravnosti promijenjen u *ispravljeno* (eng. fixed) unutar 7 dana od predaje izmjene na kodu. Usporedivši ovu tehniku s pristupom SZZ za iste projekte iz razvojne zajednice Eclipse, stopa povezivanja je povećana s 24.3% na 43.7% i pritom ima ispod 1% neispravnih veza. Naknadno je, uz pomoć samom programera-stručnjaka iz projekta Apache HTTP, ustanovljeno da neke veze mogu imati razliku u vremenu veću od 7 dana [85]. Druga inačica ove tehnike traži negaciju ispravnosti veze prethodno pronađene nekom drugom tehnikom. Pristup **D'Ambros** vezu označava kao nevaljanu ukoliko je datum prijave neispravnosti iza datuma predaje izmjene koda [89].

Obrada prirodnog jezika je tehnika novijeg datuma. Njome se može usporediti sličnost nestrukturiranog teksta iz dvaju repozitorija. Korištena je u alatu ReLink kako bi se usporedila sličnost opisa predaja i komentara neispravnosti te kako bi se predvidjele tipografske pogreške programera koji su ih ispunjavali. *Višeslojni pristup* (eng. Multi-layered approach) traži sličnosti unutar teksta u 6 slojeva uporabom tehnika obrade prirodnog jezika [90]. Prvi sloj je sličan pristupu SZZ, drugi je sličan pristupu D'Ambros, a preostali traže sličnost između fragmenata predanog programskog koda i teksta unutar izvješća o neispravnosti, između imena entiteta ili modula koji su zajednički izvješću neispravnosti i opisa predaje, ili čak traži veze koje nemaju tekstualnu sličnost.

Razvijene su još neke tehnike povezivanja. Pristup Fellegi-Sunter povezivanje temelji na 9 atributa podataka: autor i vrijeme posljednjeg komentara, autor i vrijeme pretposljednjeg komentara, autor i vrijeme predaje, ime produkta i modula te naslov predaje [91]. Taj pristup uspješno je korišten za pronalazak izostavljenih veza u repozitorijima projekata Apache i WikiMedia. Moderne tehnike povezivanja traže unapređenja u probabilističkim pristupima korištenjem raznih metoda strojnog učenja. Njihova uporaba je uočena u srodnom području *lokalizacije neispravnosti* (eng. bug localization), ali ne i u SDP-u [51]. Lokalizacija neis-

pravnosti je disciplina koja iz opisa neispravnosti nastoji pronaći dio programskog koda koji je zahvaćen tom neispravnošću.

Alati i etaloni skupova podataka

Automatizirani alati su prijeko potrebni za analizu rada, inspekciju statičkih i dinamičkih karakteristika projekata i programskog sustava koju razvijamo [9]. Njihova uporaba olakšava provedbu opsežnih zadataka te osigurava njihovu nepristranu i konzistentnu provedbu. Postoje alati za povezivanje repozitorija i alati za izračun metrika programskog koda. Alat **ReLink** razvijen je za provedbu povezivanja neispravnosti i predaja, a cilj mu je bio unaprijediti postupak pronalaskom veza koje nedostaju [92]. Alat provodi povezivanje u dva stupnja. Prvi stupanj koristi sljedeće tehnike: jednostavna pretraga, podudaranje autorstva, vremenska korelacija, te obrada prirodnog jezika, kojom uspoređuje sličnost opisa predaja i komentara neispravnosti. U drugom stupnju, alat gradi model predviđanja temeljen na vezama iz prvog stupnja te pokušava pronaći prethodno neotkrivene veze. Tehnikom obrade prirodnog jezika moguće je čak predvidjeti tipografske pogreške nastale prilikom unosa ID-a neispravnosti u opisima predaja. Usporedba rezultata ReLink alata sa skupom podataka nazvanim *temeljna istina* (eng. ground truth) [91] ukazala je kako alat daje dobre rezultate [92]. Međutim, djelotvornost povezivanja nije jednako unaprijeđena u svim projektima. Značajnije unapređenje bilo je prisutno kod manjih projekata za operativni sustav Android. Kod većih i dugotrajnijih projekata, čija je razina kvalitete podataka viša, unapređenje je bilo mnogo manje značajno. Nepristrana analiza koju su proveli drugi istraživači ukazala je na upitnu djelotvornost naprednih tehnika iz alata ReLink [93]. Osim što je unapređenje povezivanja bilo marginalno, predviđanje veza koje nedostaju dovelo je do nekih pogrešno uspostavljenih veza. Značajan doprinos rada ovih istraživača su *etaloni skupova podataka* (eng. benchmark datasets) koji sadrže ispravne veze iz deset projekata razvojne zajednice Apache.

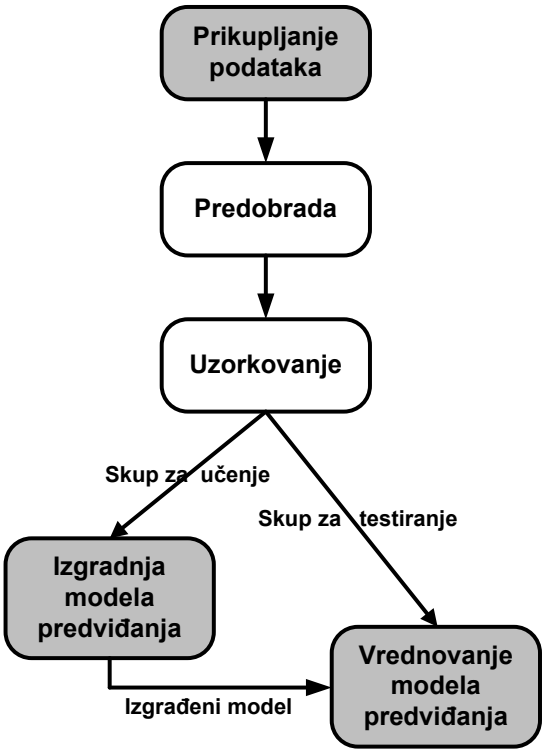
Etaloni skupova podataka prijeko su potrebni kao temelj usporedbe tehnika povezivanja. Bilo je nekoliko pokušaja uspostave takvih etalona. Bachman i grupa autora istraživali su veze koje nedostaju između neispravnosti i predaja izmjena programskog koda te ručno prikupili skup podataka temeljne istine koristeći pomoć eksperata programera koji su sudjelovali u razvoju [85]. Njihov skup podataka temeljne istine preuzeli su i nadogradili na drugim projektima i neki drugi istraživači [90, 92]. Problem je što ti skupovi podataka nisu javno dostupni. Drugačiji pristup izgradnji etalona koristili su Bissyande i grupa autora [93]. Oni su uputili kritiku na

naknadno ručno prikupljanje podataka od strane programera koji nije bio ujedno i autor samih predaja koje povezuje. Umjesto pomoći programera, oni su prikupljali podatke iz projekata koji koriste sustav za praćenje zahtjeva JIRA[¶]. Taj sustav omogućuje programerima da u trenutku predaje povežu izmjene programskog koda s neispravnošću koju su ispravljali. Koristeći tu vrijednu informaciju, zaobiđena je potreba za tehnikom povezivanja repozitorija te su izgrađeni etaloni skupova podataka za 10 projekata iz razvojne zajednice Apache. Upravo ti javno dostupni etaloni uporabljeni su u analizi koja kritizira alat ReLink [93].

[¶]https://www.atlassian.com/software/jira

2.2 Proces predviđanja programskih neispravnosti

Proces izgradnje modela za predviđanje neispravnosti prikazan je slikom 2.1 [94]. Sivom bojom označeni su dijelovi tog procesa na koje se ova disertacija više usredotočuje. Prije svega, to je proces prikupljanja podataka, korak kojim sve započinje i o čijoj ispravnosti ovise svi ostali koraci, kao što je objašnjeno u prethodnom poglavlju. Korak predobrade, odnosno pripreme podataka može uključivati rješavanje problema nedostajućih vrijednosti, *stršećih vrijednosti* (eng. outliers), *neujednačenosti podataka* (eng. data imbalance), odabira najboljeg podskupa metrika i sl. [94, 95]. Uzorkovanje podataka može se provesti jednom od sljedećih metoda: nasumična podjela ili nasumična stratificirana podjela u omjeru 2/3 : 1/3 ili 9/10 : 1/10, pri čemu veći skup služi za učenje, a manji skup služi za testiranje [95]. Postoje i metode *testiranja na jednom uzorku* (eng. leave-one-out) ili *uzorkovanja s ponavljanjem* (eng. bootstrap), koje služe za manje skupove kakve ne susrećemo u SDP-u. Model predviđanja temelji se na odabranoj metodi strojnog učenja, a potrebno je pronaći najbolji model za određene uvjete primjene. Završni korak temelji se na metrikama vrednovanja modela predviđanja, koje mogu analizirati performanse iz različitih kuteva gledanja. Izgradnja i vrednovanje modela su faze na koje neujednačenost skupova podataka može imati velik utjecaj.



Slika 2.1: Proces predviđanja neispravnosti

2.2.1 Problem neujednačenosti skupova podataka

Neujednačenost skupova podataka (eng. data imbalance) jedan je od problema koji u posljednje vrijeme zaokuplja sve veću pažnju u području strojnog učenja, budući da je problem prisutan u većini područja primjene dubinske obrade podataka [96]. Poteškoću predstavlja kada je jedna klasa podataka u velikoj mjeri brojnija od druge, a podešavanje parametara modela predviđanja temelji se na točnosti predviđanja. Upravo ta manjinska klasa podataka najčešće je od većeg interesa jer predstavlja neki rijedak događaj koji je važno otkriti. Ukoliko je ta klasa značajno manje zastupljena, moguće je postići visoku točnost čak i ako se sve uzorke svrsta u većinsku klasu. To je slučaj neujednačenih skupova podataka, a čest je u predviđanju programskih neispravnosti, naročito kod sustava s niskom razinom tolerancije na neispravnosti, poput medicinskih, aeronautičkih ili telekomunikacijskih sustava [37]. U takvim sustavima, cijena neotkrivene neispravnosti mnogo je veća od cijene uzalud tražene neispravnosti. Međutim, različita cijena pogrešnog predviđanja nije uključena u uobičajenim modelima predviđanja [97]. S druge strane, ujednačeni skupovi podataka su oni kod kojih su sve klase podataka približno jednako zastupljene. Postoje brojne tehnike za pripremu podataka i učenje iz neujednačenih skupova podataka, a mogu se svrstati u sljedeće kategorije [98]:

- Metode pripreme podataka *tehnikama za uzorkovanje* (eng. sampling techniques);
- *Metode učenja osjetljivih na cijenu* (eng. cost-sensitive learning methods);
- *Jezgrene metode učenja* (eng. kernel-based learning);
- *Aktivne metode učenja* (eng. active learning methods).

Budući da tradicionalno korištene metrike vrednovanja opisane u poglavlju 2.2.3, poput točnosti, nisu prikladne, važnost je dana i pronalasku odgovarajućih metrika vrednovanja [99, 100]. Posljednja istraživanja uključuju i prilagođavaju nove algoritme, poput metode slučajne šume (eng. random forest) [101], ili čak optimizacijskih heurističkih algoritama u klasifikaciji neujednačenih skupova podataka [102, 103]. Međutim, iako su tehnike iz pojedinih skupina metoda međusobno uspoređivane [104], a neka istraživanja ukazuju na njihovu upitnu djelotvornost kod svih modela predviđanja [105], još ne postoji sustavna usporedba učinkovitosti svih tehnika. Stoga je jedan od ciljeva ove disertacije predstavljanje okvira za usporedbu metoda strojnog učenja u različitim uvjetima neujednačenosti skupova podataka.

Većina studija u početku nije analizirala razinu neujednačenosti skupova podataka. Uvođenjem šireg raspona mjera za vrednovanje rezultata modela predviđanja dobiveni su neki obeshrabrujući rezultati [116]. Otkriveno je da upravo neujednačenost skupova podataka može uzro-

Tablica 2.1: Granična vrijednost neispravnosti u srodnoj literaturi

Ref.	Domena primjene	**GVN**	**% SNP**	Motivacija za odabrani GVN
[106]	projekt otvorenog koda	$<$ medijan	43%	radi postizanja ujednačenih skupova
[107]	projekt otvorenog koda	$<$ medijan	nepoznato	radi postizanja ujednačenih skupova
[108]	registarski sustav	< 20	20%	korišten od strane drugih studija
	za telekomunikaciju	< 5	66%	određen od strane visokog menadžmenta
[40]	telekomunikacijski sustav	< 10	40%	granica odabrana iskustveno nakon diskusije sa projektnim inženjerima
[109]	medicinski sustav	< 10	4%	jedan red veličine manje od najvećeg broja
		< 4	1%	neispravnosti, ali neobjašnjene motivacije
[110]	radarski sustav vojne primjene	< 4	27%	'projektni inženjeri su smatrali ovaj GVN zadovoljavajućom za vođenje razvoja'
[111]	telekomunikacije	< 3	14%	predodređena veličina grupa
[40]	navigacijski sustav	< 1	40%	vrlo visoki zahtjev za kvalitetu koji ne toleriraju postojanje neispravnosti
[112]	projekt otvorenog koda	< 1	nepoznato	nije objašnjena
[113]	telekomunikacije	< 1	$< 6\%$	nije objašnjena
[114]	operativni sustav	< 1	nepoznato	nije objašnjena
[115]	sustav za menadžment	< 1	47%	nije objašnjena

kovati takve rezultate [117, 118]. Međutim, pojedine studije su manipulirale razinom neujednačenosti skupova tako što su mijenjale graničnu vrijednost neispravnosti (GVN), koja prepoznaje dvije klase podataka: *sklone neispravnostima* (SNP, eng. fault prone) i *nesklone neispravnostima* (NNP, eng. non-fault prone). Riječ je o strateškoj odluci, koju visoki menadžment nerijetko donosi na temelju određenog iskustva prilikom definicije strategije verifikacije programskog koda. Što je veća granična vrijednost neispravnosti, to je udio manjinske klase (SNP) manji, a razina neujednačenost veća. U tablici 2.1 silazno su poredani brojevi koji su se koristili za graničnu vrijednost neispravnosti (GVN) u srodnoj literaturi. Prikazan je i posljedični udio manjinske klase (%SNP) te domena primjene sustava iz kojeg podaci potječu. Udio manjinske klase (%SNP) varira od vrlo malih vrijednosti u medicinskim sustavima do relativno ujednačenih vrijednosti od oko 50% u sustavima za telekomunikaciju i projektima otvorenog koda.

Razina neujednačenosti skupova podataka izravno ovisi o odabiru broja za granične vrijednosti neispravnosti, a interesantno je promotriti motivaciju u pozadini tog odabira. Objašnjenje nije ponuđeno u većini slučajeva kada je broj za graničnu vrijednost neispravnosti postavljen na 1. Mnogi smatraju da nije niti potrebno ponuditi objašnjenje ukoliko se za module SNP uzmu u obzir svi koji imaju barem jednu neispravnost. Međutim, jedino objašnjenje koje stoji iza tog odabira je vrlo visok zahtjev kvalitete proizvoda, koji ne smije sadržavati neispravnosti. Iz tablice 2.1 uočeno je da iskustvo inženjera ili visokog menadžmenta najčešće motivira odabir broja za GVN koji je veći od 1. Jedini primjer u kojem je motivacija za odabrani GVN povezana s razinom neujednačenosti su [106] i [107]. Tada se ne koristi predefinirani broj neispravnosti, već se uzima medijan broja neispravnosti kako bi se dobio ujednačen skup podataka. Studije koje GVN postavljaju na vrijednost veću od 1 dolaze iz industrijskog okruženja s niskom tolerancijom na neispravnosti. Njihova motivacija je pretežno proizašla iskustveno, iz diskusije projektnih inženjera. Upravo ta činjenica motivira i ovu disertaciju da manipulacijom GVN-a istraži utjecaj neujednačenosti skupova podataka na ponašanje modela predviđanja.

2.2.2 Odabir metode predviđanja

Postoji velik broj metoda za izgradnju modela za predviđanje koje se temelje na metodama strojnog učenja. Predviđanje programskih neispravnosti unutar programskih modula može biti regresijsko ili klasifikacijsko. Regresijom se predviđa broj neispravnosti po modulu, a klasifikacijom se moduli svrstavaju u dvije kategorije (SNP i NNP) ili u više njih. Iz gledišta osiguranja kvalitete i dodjele sredstava na module koji posjeduju neispravnosti, klasifikacija modula u dvije kategorije je prirodniji, a u srodnoj literaturi i najčešći odabir. Metode strojnog učenja koji se pri tome koriste mogu se svrstati u sljedeće kategorije [44]:

1. Statistički algoritmi za klasifikaciju:
   1.1. *Linearna diskriminantna analiza* (eng. linear discriminant analysis);
   1.2. *Kvadratna diskriminantna analiza* (eng. quadratic discriminant analysis);
   1.3. *Logistička regresija* (eng. logistic regression);
   1.4. *Naivan Bayesov klasifikator* (eng. naive Bayes);
   1.5. *Bayesova mreža* (eng. Bayesian networks);
   1.6. *Regresija najmanjih kuteva* (eng. least-angle regression).
2. Metode najbližih susjeda:
   2.1. *k-najbližih susjeda* (eng. k-nearest neighbor);

2.2. *k-zvijezda* (eng. k-star).

3. Neuronske mreže:
   3.1. *Višeslojni perceptron* (eng. multi-layer perceptron);
   3.2. *Mreža zasnovana na radijalnim baznim funkcijama* (eng. radial basis function network).
4. Metode zasnovane na potpornim vektorima:
   4.1. *Metoda potpornih vektora* (eng. support vector machine, SVM);
   4.2. *Lagrangian-ova metoda potpornih vektora* (eng. lagrangian SVM);
   4.3. *Metoda potpornih vektora najmanjim kvadratima* (eng. least squares SVM).
5. Metode zasnovane na stablima odlučivanja:
   5.1. *Stablo odlučivanja C4.5* (eng. C4.5 decision tree);
   5.2. *Stablo za klasifikaciju i regresiju* (eng. classification and regression tree);
   5.3. *Stablo izmjeničnog odlučivanja* (eng. alternating decision tree).
6. *Grupne metode* (eng. ensemble methods):
   6.1. *Slučajna šuma* (eng. random forest);
   6.2. *Model logističkih stabala* (eng. logistic model tree);
   6.3. *Rotirajuća šuma* (eng. rotation forest);
   6.4. *Višeciljno genetičko programiranje* (eng. multi-objective genetic programming).

Logistička regresija, koja je detaljnije opisana u poglavlju 2.3.1, poznata je kao pouzdana i robusna metoda. Rangirana je među boljim metodama za SDP u nekoliko komparativnih studija [42, 44, 100, 106, 107, 119, 120]. Prema nekim istraživanjima, to je najbolja statistička tehnika koju sva buduća istraživanja svakako trebaju uzeti u obzir [121]. Sve navedeno poslužilo je kao motivacija za uključivanje te metode u istraživanje ove disertacije.

Metoda potpornih vektora, koja je opisana u poglavlju 2.3.2, pokazala se kao veoma uspješan algoritam u mnogim stvarnim primjenama, čak i u neujednačenim skupovima. Međutim, pokazano je da njihove performanse dramatično opadaju u uvjetima visoke razine neujednačenosti [122]. Upravo stoga, bilo bi vrlo korisno otkriti točno pri kojoj razini neujednačenosti ova metoda prestaje biti svrsishodna.

Prednosti metode slučajne šume su da je robusna na stršeće vrijednosti i šum, brža je od grupnih metoda zasnovanih na ponavljajućem uzorkovanju *bagging* i *boosting*, točnija je od metoda poput *adaboost*, a s većim brojem stabala odlučivanja manja je vjerojatnost pojave *prenaučenosti* (eng. overfitting) [123]. Metoda slučajne šume korištena je u iscrpnom i sustavnom eksperimentu koji je analizirao performanse nekih metoda strojnog učenja u domeni neujed-

načenih skupova podataka [124]. Koristeći deset neujednačenih skupova podataka iz različitih domena stvarne primjene, slučajna šuma je uspoređena s naivnim Bayesovim klasifikatorom, metodom potpornih vektora, *K-najbližih susjeda* i *stablom odlučivanja C4.5*. Rezultati vrednovani mjerom *površine ispod krivulje* (eng. Area Under Curve, AUC) ROC (eng. Receiver Operating Curve) i uspoređeni Kolmogorov-Smirnovljevim statističkim testom govorili su u prilog korištenju upravo te metode. Detaljniji opis metode slučajne šume nalazi se u poglavlju 2.3.4.

Rotirajuća šuma jedna je od najnovijih metoda strojnog učenja, a predstavljena je u poglavlju 2.3.5. Njezini autori, Rodriguez i grupa autora [125], usporedili su je sa *bagging*, *adaboost* i slučajnom šumom u kontekstu 33 nasumično odabrana etalona skupova podataka iz repozitorija za strojno učenje UCI. Ti podaci sadržavali su od 4 do 69 nezavisnih varijabli, između 58 i 20,000 uzoraka te između 2 i 26 klasa. Uporabljen je statistički test značajne razlike među grupama rezultata izraženih mjerom točnosti te je rotirajuća šuma ostvarila obećavajući uspjeh od 84 pobjeda i svega 2 poraza, pri čemu su se pobjede računale za statistički signifikantno bolje rezultate. Slična studija usporedila je metode *bagging*, *random subspaces*, slučajna šuma i rotirajuća šuma na 43 skupa podataka iz istog repozitorija [126]. Rezultati su također ukazivali da je rotirajuća šuma najbolja metoda strojnog učenja. Koliko je poznato, uporaba ove metode još je relativno neistražena u području SDP-a. Jedina komparativna studija koja je to učinila usporedila je sljedeće metode: stablo odlučivanja *C4.5*, *SMO*, rotirajuća šuma, *bagging*, *adaBoost*, slučajna šuma i *DBScan* [127]. Slučajna šuma je pritom postigla najbolje rezultate u gotovo svih osam mjera vrednovanja koje su korištene. Zamjerka toj studiji je izostanak opisa procesa prikupljanja podataka i razvojnog okruženja iz kojeg potječu.

2.2.3 Vrednovanje modela

Vrednovanje modela zasniva se na tablici zabune [128]. Svi mogući ishodi točnog i netočnog svrstavanja analiziranih uzoraka u klase za binarnu klasifikaciju definirani su tablicom 2.2. U ovoj studiji, uzorci su programski moduli, a klase su sklonost neispravnostima (SNP, pozitivno, 1) i nesklonost neispravnostima (NNP, negativno, 0). Ishodi koji su mogući za binarnu klasifikaciju se pritom računaju kao:

- Točno pozitivno (eng. True Positive, TP) za svaki modul kojem je predviđena sklonost (1) i on je stvarno posjeduje (1);
- Netočno pozitivno (eng. False Positive, FP) za svaki modul kojem je predviđena sklonost

(1), a on je ne posjeduje (0) - često nazivan *pogreškom tipa 1* (eng. type I error);

- Netočno negativno (eng. False Negative, FN) za module kojima nije predviđena sklonost (0), a zapravo je posjeduju (1) - često nazivan *pogreškom tipa 2* (eng. type II error);
- Točno negativno (eng. True Negative, TN) za module kojima nije predviđena sklonost (0) i oni je stvarno ne posjeduju (0).

Tablica 2.2: Tablica zabune za binarnu klasifikaciju

Stvarno Stanje:	Predvideno:	
	Pozitivan (1)	Negativan (0)
Pozitivan (1)	**TP**	FN
Negativan (0)	FP	**TN**

Istraživači najčešće priželjkuju mjeru vrednovanja čija će vrijednost biti u rasponu [0, 1] kako bi usporedili dobrotu modela predviđanja. Prebrojavanjem pojava za četiri moguća ishoda iz tablice zabune, moguće je izračunati nekoliko mjera za vrednovanje uspješnosti modela za predviđanje. U tablici 2.3 prikazane su popularne i najčešće korištene mjere vrednovanja.

Točnost

Točnost (eng. accuracy, Acc) predstavlja udio ispravno klasificiranih modula u ukupnom broju modula. Često se koristi kao pokazatelj općenite primjerenosti nekog modela za predviđanje. Međutim, ova mjera može navesti na pogrešan zaključak u slučaju visoke razine neujednačenosti dviju klasa [128]. Primjerice, za slučaj da skup podataka posjeduje jako malo modula sklonih neispravnostima (SNP) (npr. manje od 10%), model predviđanja može neispravno predvidjeti da su svi moduli neskloni neispravnostima (NNP) i postići veliku točnost (veću od 90%). Stoga je jako važno koristiti i druge mjere vrednovanja.

Udio ispravnog klasificiranja

Osjetljivost (eng. sensitivity) se definira kao *udio točno pozitivnih modula* (eng. true positive rate, TPR) u svim modulima koji su stvarno pozitivni. Računa se kao postotak ispravno klasificiranih modula sklonih neispravnostima u ukupnom broju modula sklonih neispravnostima (SNP). Ova mjera je važna jer se za njezinu visoku vrijednost zna da je uspješno pronađena većina traženih modula.

Tablica 2.3: Mjere vrednovanja uspješnosti modela za predviđanje

Naziv	Oznaka	Definicija
Točnost	Acc	$\frac{TP+TN}{TP+FP+TN+FN}$
Osjetljivost	TPR	$\frac{TP}{TP+FN}$
Lažni alarm	FPR	$\frac{FP}{FP+FN}$
Specifičnost	TNR	$\frac{TN}{TN+FP}$
Preciznost	PR	$\frac{TP}{TP+FP}$
Težinska prosječna točnost	Ave	$W(TPR)+(1-W)(TNR)$
F-mjera	FM	$2 \cdot \frac{TPR \cdot PR}{TPR+PR}$
Geometrijska sredina	GM	$\sqrt{TPR \cdot TNR}$
Površina ispod krivulje ROC	AUC	$\int_0^1 ROC_{curve}$

Lažni alarm (eng. false alarm, fall-out) se definira kao udio netočno pozitivnih modula (eng. false positive rate, FPR) u ukupnom broju neispravno klasificiranih modula. Računa se kao postotak neispravno klasificiranih modula sklonih neispravnostima u ukupnom broju neispravno klasificiranih modula. Često se koristi kao dopuna osjetljivosti jer visoka razina lažnih alarma znači da se često griješi u pronalasku traženih modula.

Specifičnost (eng. specificity) se definira kao udio točno negativnih modula (eng. true negative rate, TNR) u svim modulima koji su stvarno negativni. Računa se kao postotak ispravno klasificiranih modula nesklonih neispravnostima u ukupnom broju modula nesklonih neispravnostima (NNP). Ova mjera je manje važna za SDP, budući da je cilj pronalazak modula koji su SNP. Za slučaj da je skup podataka neujednačen, mnogi modeli predviđanja skloniji su visokoj specifičnosti.

Preciznost (eng. precision, PR) je poznata i pod nazivom *pouzdanost* (eng. confidence). Računa se kao postotak ispravno klasificiranih modula sklonih neispravnostima u ukupnom broju modula koji su predviđeni kao skloni neispravnostima.

Prosječna točnost

Težinska prosječna točnost klasificiranja (eng. weighted average classification accuracy, Ave) je mjera koja kombinira točnost manjinske klase (TPR) i točnost većinske klase (TNR). Upotrebom težinskog faktora W može se dati veća važnost jednoj ili drugoj klasi podataka. Ova mjera je predložena 2013. godine i korištena prvenstveno kao funkcija dobrote za model predviđanja zasnovan na genetskom programiranju [102].

F-mjera (eng. F-measure, FM), poznata i pod nazivom *pokazatelj* F_1 (eng. F_1 score). Računa se kao harmonijska srednja vrijednost između osjetljivosti (TPR) i preciznosti (PR).

Geometrijska sredina (eng. geometric mean, GM) također kombinira točnost manjinske klase (TPR) i većinske klase (TNR). Računa se kao geometrijska sredina tih dviju mjera i smatra se mnogo robusnijom mjerom od točnosti i F-mjere [129]. Naročito se smatra korisnom u klasifikaciji neujednačenih skupova podataka jer više penalizira odstupanje točnosti u samo jednoj klasi [130].

Površina ispod krivulje

Postoji nekoliko mjera koje se računaju kao *površina ispod krivulje* (eng. area under curve, AUC), a najčešće je korištena krivulja ROC (eng. receiver operating curve). Ova mjera kombinira osjetljivost (TPR) i učestalost lažnog alarma (FPR), a njezin se izračun temelji na vjerojatnosti koju model predviđanja daje kao izlaznu vrijednost. Mijenjanjem vrijednosti granične vjerojatnosti iznad koje se prihvaća klasifikacija modela računaju se uređeni parovi (TPR, FPR) koji čine krivulju ROC. Idealan slučaj je točka (1, 0), a realan slučaj je da krivulja kreće od (0,0) i završava u (1,1), a integral takve krivulje je mjera koju se označava s AUC. Za vrijednost 0.5 predviđanje se smatra jednakim nasumičnom pogađanju, a vrijednost 1 je idealna. U domeni predviđanja za SDP, ovo je i najpopularnija mjera [94]. Međutim, u prisutnosti iznimno neujednačenih skupova podataka pokazalo se da ova mjera ne daje suvisle rezultate [131]. Postoje i druge mjere koje se računaju kao površina ispod krivulje, poput *krivulje troška* (eng. cost curve) ili *krivulje izdizanja* (eng. lift chart) [100]. Međutim, mjere koje iziskuju poznavanje troška neispravne klasifikacije nerijetko je vrlo teško odrediti.

2.3 Metode strojnog učenja

U prethodnom poglavlju izdvojeno je pet metoda strojnog učenja, a one su: logistička regresija, naivan Bayesov klasifikator, metoda potpornih vektora, slučajna šuma i rotirajuća šuma. One su odabrane među brojnim metodama strojnog učenja zbog pozitivnih iskustava u različitim domenama njihove primjene. Neke od tih metoda poznate su već duže vrijeme. Korištene su u brojnim problemima klasifikacije, a neke su novije i pokazuju iznimno obećavajuće rezultate. Sve su metode korištene barem jednom u istraživanjima za SDP i postizale su dobre rezultate. Za neke se ustvrdilo da imaju dobro ponašanje u slučaju neujednačenih skupova podataka, a za neke je poznato da im se performanse narušavaju u takvim uvjetima. No niti za jednu nije utvrđeno u kojim točno razinama neujednačenosti daju dobre, najbolje ili loše rezultate i može li se odrediti takva zavisnost u skupovima podataka za SDP. U ovom poglavlju predstavljen je mehanizam učenja, odnosno podešavanja parametara i primjene tih modela.

2.3.1 Logistička regresija

U statistici, logistička regresija poznata je kao metoda modeliranja za predviđanje vjerojatnosti ishoda nekog događaja. Njezina najčešća svrha je modeliranje *a posteriori* vjerojatnosti za dva moguća ishoda, odnosno za dvije klase, uporabom linearne funkcije od ulaznih varijabli x. Logističkom funkcijom osigurava se da izlazna vrijednosti može poprimiti samo vrijednosti između nula i jedan [132]. Navedena karakteristika čini ju privlačnom za izgradnju modela za predviđanje programskih neispravnosti koji poznaje samo dva moguća ishoda, a često se koristi i u medicini [133]. Logistička regresija srodna je drugim statističkim tehnikama analize, ali pruža viši stupanj fleksibilnosti i robusnosti. Njezina robusnost proizlazi iz činjenice da ne polazi od pretpostavke linearne ovisnosti između ulaznih i izlaznih varijabli, niti normalne distribucije ili jednakih varijanci između ulaznih varijabli [134]. Upravo stoga, logistička regresija je pogodna za razne primjene u domeni programskog inženjerstva čiji podaci najčešće nisu normalne, već zakrivljene distribucije, koje nerijetko sadrže stršeće vrijednosti, a ponekad imaju i nedostajuće vrijednosti. Njezin model ima sljedeću formu [134]:

$$ln\left(\frac{\widehat{Y}_j}{1-\widehat{Y}_j}\right) = u \tag{2.1}$$

pri čemu je $\widehat{Y}_j$ procjena vjerojatnosti u rasponu [0, 1] da j-ti uzorak pripada jednoj od klasa,

a u je uobičajena linearna regresija od ulazne varijable x:

$$u = A + Bx \tag{2.2}$$

uz konstantu A, koeficijent B i ulaznu vrijednost x na kojoj se temelji predviđanje. Jednostavnom pretvorbom, logistička jednadžba dana u izrazu 2.1 može se uporabiti za izračun tražene vjerojatnosti iskazane izrazom [134]:

$$\widehat{Y}_j = \frac{e^u}{1+e^u} \tag{2.3}$$

Izlazna vrijednost jednadžbe 2.3 je vjerojatnost događaja. Za potrebe predviđanja ishoda potrebno je odrediti graničnu vrijednost vjerojatnosti iznad koje se smatra da je neki ishod dovoljno vjerojatan. Ta granica uobičajeno se postavlja na 0.5, ali postoje i druge metode za njezino podešavanje, koje mogu uzeti u obzir različitu cijenu pogrešnog predviđanja između FP i FN ishoda klasifikacije [37].

Za slučaj da postoji više od jedne ulazne varijable, koristi se multivarijantni model logističke regresije. Njezin model zasnovan je na izrazu 2.3 te glasi [134]:

$$\widehat{Y}_j\left(m_1, m_2, ...m_n\right) = \frac{e^{C_0+C_1m_1+...+C_nm_n}}{1+e^{C_0+C_1m_1+...+C_nm_n}} \tag{2.4}$$

pri čemu atribut m_i predstavlja i-tu ulaznu varijablu (za SDP to mogu biti metrike programskog koda) koja je uključena u model predviđanja, C_i predstavlja regresijski koeficijent i-te varijable, C_0 je slobodni član, a $\widehat{Y}_j$ je izlazna vjerojatnost.

Metodom *maksimalne vjerodostojnosti* (eng. maximum likelihood), odnosno izračunom *logaritma vjerodostojnosti* (eng. natural log likelihood, NNL) procjenjuju se vrijednosti koeficijenata. Logaritam vjerodostojnosti računa se između stvarnih (Y_j) i predviđenih ($\widehat{Y}_j$) vrijednosti za svih N promatranih uzoraka izrazom [134]:

$$NNL = \sum_{j=1}^{N}\left[Y_j ln\left(\widehat{Y}_j\right) + \left(1-Y_j\right) ln\left(1-\widehat{Y}_j\right)\right] \tag{2.5}$$

Metoda ima za cilj pronaći najbolju linearnu kombinaciju ulaznih varijabli kako bi vjerodostojnost izlaznih vrijednosti bila maksimizirana. Metoda maksimalne vjerodostojnosti je iterativna procedura koja počinje s nasumično odabranim vrijednostima koeficijenata za svaku ulaznu varijablu. Potom se određuje smjer i veličina izmjena njihovih vrijednosti koje će dovesti

do najveće vjerodostojnosti za promatrane uzorke. Model predviđanja temeljen na izračunatim vrijednostima koeficijenata testira se i ponovo se određuje smjer i veličina izmjena. Procedura se ponavlja sve dok koeficijenti ne konvergiraju prema stabilnim vrijednostima. Metoda maksimalne vjerodostojnosti daje one vrijednosti koeficijenata nezavisnih varijabli za koje će izlazna vjerojatnost najbolje odgovarati klasama iz podataka za učenje [132].

2.3.2 Naivan Bayesov klasifikator

Naivan Bayesov klasifikator je statistički algoritam koji dodjeljuje vjerojatnost klasifikaciji uzoraka u pojedinu klasu. Vjerojatnosti se određuju frekvencijskom interpretacijom i promatranjem svakog atributa nezavisno. Pretpostavka uvjetne nezavisnosti među atributima naziva se pretpostavkom "naivnosti", a uvedena je radi pojednostavljenja izračuna [132]. Naivan Bayesov klasifikator se zasniva na Bayesovom teoremu pa je po Thomasu Bayesu i dobila ime. Mnoge studije su pokazale da se naivnim Bayesovim klasifikatorom dobivaju dobri rezultati, kao i metodom stabla odlučivanja ili metodom neuronske mreže. Naivan Bayesov klasifikator karakterizira visoka točnost i brzina čak i u radu s velikim skupovima podataka [135].

Algoritam uključuje sljedeće korake prilikom podešavanja parametara modela [135]:

(1) Za svaki uzorak M iz skupa za učenje X koji pripada jednoj klasi C_i od ukupno k klasa, računa se najveća *a posteriori* vjerojatnost $P(C_i|M) > P(C_j|M)$ za svaki $1 \leqslant j \leqslant k,\ j \neq i$ koristeći Bayesov teorem:

$$P(C_i|M) = \frac{P(M|C_i)P(C_i)}{P(M)} \tag{2.6}$$

(2) Budući da je $P(M)$ konstantan za sve klase, potrebno je maksimizirati $P(M|C_i)P(C_i)$. Ako je vjerojatnost klasa *a priori* nepoznata, uobičajeno je pretpostaviti da su klase jednako vjerojatne $P(C_1) = P(C_2) = ...P(C_k)$ pa se maksimizira $P(M|C_i)$. U domeni SDP, postoje samo dvije klase, a njihove vjerojatnosti mogu se izračunati tako da se izračuna njihova učestalost u skupu za učenje.

(3) Računski bi bilo prezahtjevno izračunati $P(M|C_i)$ za skupove podataka koji imaju velik broj atributa n. Stoga se uvodi naivna pretpostavka uvjetne nezavisnosti među metrikama, te se račun svodi na umnožak:

$$P(M|C_i) = \prod_{j=1}^{n} P(m_j|C_i) \tag{2.7}$$

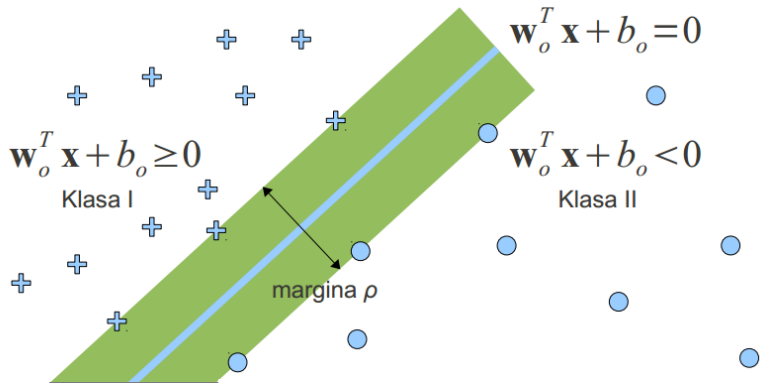
pri čemu atribut m_i predstavlja i-tu ulaznu varijablu (za SDP to mogu biti metrike programskog koda). Uvjetne vjerojatnosti $P(m_j|C_i)$ se jednostavno izračunaju iz skupa za učenje. Za kategoričke vrijednosti metrike m_j, $P(m_j|C_i)$ je učestalost klasa C_i koje imaju vrijednost m_j. Za kontinuirane vrijednosti metrike m_j, $P(m_j|C_i)$ računa se Gaussova razdioba, odnosno srednja vrijednost i odstupanje koju metrika m_j ima za klasu C_i.

(4) Da bi se predvidjela klasa kojoj pripada uzorak M, vjerojatnost $P(M|C_i)P(C_i)$ se računa za svaku klasu C_i. Konačna odluka algoritma za klasifikaciju jednaka je onoj klasi koja ima najveću vjerojatnost $P(M|C_i)P(C_i)$, odnosno za koju vrijedi

$$P(M|C_i)P(C_i) > P(X|C_j)P(C_j), \quad za\ svaki\ 1 \leqslant j \leqslant k,\ j \neq i \tag{2.8}$$

2.3.3 Metoda potpornih vektora

Metoda potpornih vektora (eng. support vector machine) je algoritam za klasifikaciju koji nelinearnim preslikavanjem prevodi originalne podatke u višu dimenziju. Unutar nove dimenzije, traži optimalnu linearnu *hiperravninu* (eng. hyperplane) koja će dijeliti dvije klase podataka. Ideja u pozadini ovog algoritma je, da se uporabom odgovarajuće tehnike nelinearnog preslikavanja podaci mogu podići u dovoljno visoku dimenziju u kojoj će biti odvojivi hiperravninom. Metoda potpornih vektora pronalazi tu ravninu uz pomoć potpornih vektora i margina koje oni definiraju. Začeci ovog algoritma pronalaze se 1960-ih godina u radovima V. Vapnika i A. Chervonenkisa, ali njegova šira primjena započela je 30-ak godina kasnije [136]. Razlog tomu je računski zahtjevan proces podešavanja parametara modela, koji i danas može biti spor, čak i kod najbržih konfiguracija. Međutim, modeliranjem nelinearnih uvjeta odlučivanja postiže se visoka točnost i manja *prenaučenost* (eng. overfitting) [135].



Slika 2.2: Hiperravnina i rub metode potpornih vektora

Hiperravnina se može definirati izrazom [135]:

$$\mathbf{W} \cdot \mathbf{M} + b = 0 \tag{2.9}$$

pri čemu $\boldsymbol{W}$ predstavlja vektor težina za svaku od n metrika, a b je skalar koji se naziva pomak. Za klase definirane kao $y_1 = 1$ i $y_2 = -1$, pripadnost određenoj klasi definira se s obzirom na položaj promatranog uzorka izrazom [135]:

$$y_i \left(w_o + w_1 x_1 + ... + w_n x_n\right) \geqslant 1, \ \ \forall i \tag{2.10}$$

Rub (eng. margin) hiperravnine je najmanja udaljenost hiperravnine od najbližeg uzorka iz skupa za učenje [135]. Udaljenost se računa kao euklidska udaljenost, a rubna hiperravnina je ravnina paralelna s hiperravninom čija udaljenost je uvijek jednaka rubu. Točke koje se nalaze točno na rubu su one koje je najteže klasificirati, a nazivaju se potpornim vektorom. Optimalna hiperravnina se naziva i *hiperravnina maksimalne razlučivosti* (eng. maximum marginal hyperplane). Maksimalna razlučivost postiže se tako da se pronađe hiperravnina čiji rub ima najveću moguću vrijednost. Slika 2.2 prikazuje primjer linearno razdvojivih klasa te definiciju hiperravnine maksimalne razlučivosti. Algoritam je završio s podešavanjem parametara modela čim pronađe potporne vektore, a njegova složenost je okarakterizirana brojem potpornih vektora [132]. Kada se odrede potporni vektori i hiperravnina maksimalne razlučivosti, podešavanje parametara modela temeljenog na metodi potpornih vektora je završeno. Hiperravnina maksimalne razlučivosti linearno razdvaja pripadnike dviju klasa. Koristeći Lagrangianovu jednadžbu, hiperravnina maksimalne razlučivosti može se raspisati kao:

$$d\left(X^T\right) = \sum_{i=1}^{l} y_i \alpha_i X_i X^T + b_0 \tag{2.11}$$

pri čemu y_i je oznaka klase za potporni vektor X_i, X^T je testni uzorak, α_i i b_0 su numerički parametri koji se odrede optimizacijom algoritma za podešavanje parametara modela temeljenog na metodi potpornih vektora, a l je broj potpornih vektora.

Opisani proces koristi se za klasifikaciju linearno odvojivih uzoraka. Za podatke koji nisu linearno odvojivi, potrebno je povečati dimenzionalnost te ponovo računati hiperravninu najveće razlučivosti [135]. Povećanje dimenzionalnosti podataka zahtijeva računski izuzetno zahtjevne matrične umnoške. Zato se na originalne podatke primjenjuju matematički ekvivalentne *jez-*

grene funkcije (eng. kernel function) $K(X_i, X_j)$:

$$K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j) \tag{2.12}$$

pri čemu $\phi()$ predstavlja funkciju transformacije u višu dimenziju. Na taj se način svi izračuni rade u dimenziji originalnih podataka, što može biti mnogo niže dimenzionalnosti nego li bi isti izračun bio bez da se koriste jezgrene funkcije. Sljedeće jezgrene funkcije prihvaćene su u izgradnji metode potpornih vektora [135]:

$$\textbf{Polinomna jezgrena funkcija stupnja } h: \quad K(X_i, X_j) = \left(X_i \cdot X_j + 1\right)^h \tag{2.13}$$

$$\textbf{Gaussova radijalna jezgrena funkcija}: \quad K(X_i, X_j) = e^{-||X_i - X_j||^2/2\sigma^2} \tag{2.14}$$

$$\textbf{Sigmoidna jezgrena funkcija}: \quad K(X_i, X_j) = tanh\left(\kappa X_i \cdot X_j - \delta\right) \tag{2.15}$$

2.3.4 Slučajna šuma

Slučajna šuma (eng. random forest) je *grupni* (eng. ensemble) klasifikacijski algoritam, koji je 2001. godine predložio Leo Breiman [123]. Algoritam se zasniva na uvođenju nasumičnosti u podjeli uzoraka i atributa za velik broj nezavisnih *stabala odlučivanja* (eng. decision tree). Svako stablo odlučivanja je jedan klasifikacijski algoritam koji dobije drugačiji skup za učenje. Kao i u ostalim grupnim metodama, svaki član grupe ima jedan glas u većinskom donošenju konačne odluke klasifikacije. Na taj je način umanjena pogreška u predviđanju, te minimiziran utjecaj stršećih vrijednosti, a performanse su unaprijeđene. Algoritam metode slučajne šume prikazan je slikom 2.3. Uz proizvoljno stablo označeno kao T_i te ukupan broj stabala K, algoritam uključuje sljedeće korake [132]:

(1) Svakom stablu T_i dodijeli se podskup metrika čiji broj može biti između 1 i $\sqrt{n}$ iz skupa za učenje;

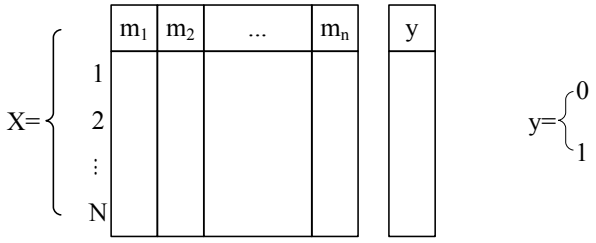
(2) Iz podskupa se uzorkuje 2/3 uzoraka za učenje i 1/3 uzoraka za procjenu pogreške uporabom *bootstrap* metode;

(3) Iterativno se grade stabla koristeći *CART* (eng. Classification And Regression Tree) metodologiju bez *podrezivanja* (eng. pruning);

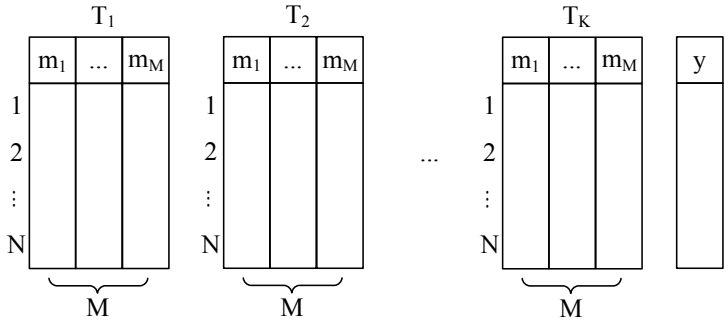
(4) Koristeći skup za procjenu pogreške provede se individualno i grupno usrednjavanje pogreške stabala;

(5) Izlazna vrijednost grupne klasifikacije temelji se na većinskom glasanju svih stabala u šumi.

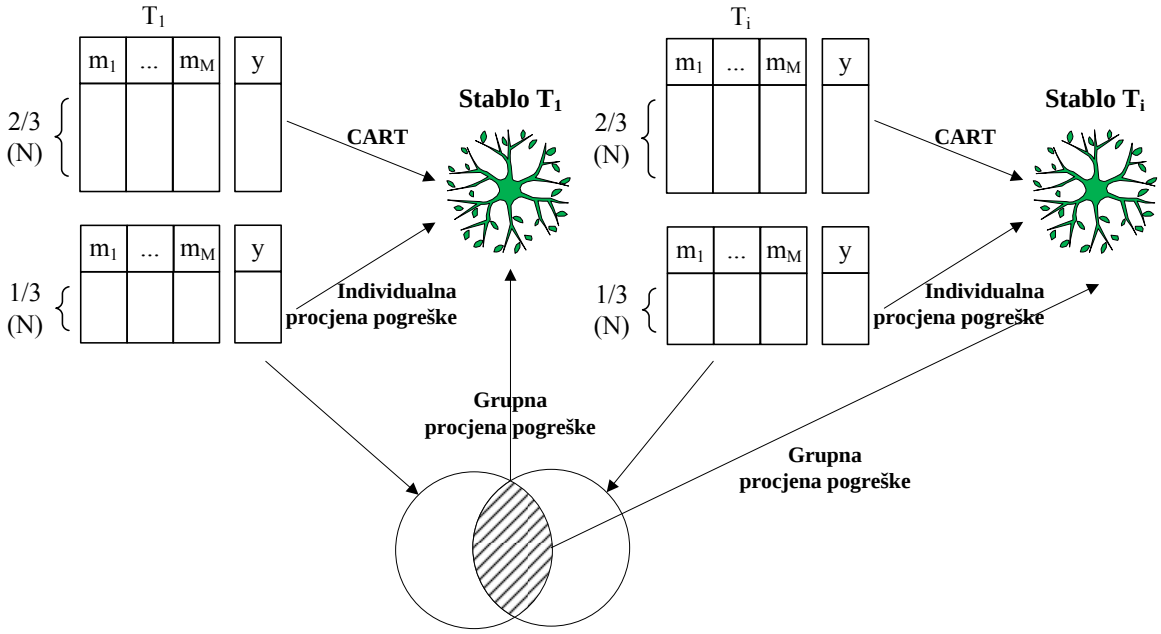
- Skup za učenje X sadrži *n* nezavisnih varijabli (metrika) i *y* kao zavisnu varijablu u *N* slučajeva



1. Svako stablo T_i ($1 \le i \le K$) dobiva nasumično odabrani podskup metrika veličine M ($1 \le M \le \sqrt{n}$)



- Svako stablo T_i dobiva 2/3 slučajeva za učenje i 1/3 za procjenu pogreške, odabranih bootstrap metodom
- Metoda učenja je CART metodologija bez *podrezivanja* (eng. pruning)
- Pogreške se procjenjuje individualno i grupno za podskup stabala



- Provodi se usrednjavanje pogreške nad svim stablima
- Konačna klasifikacijska odluka se temelji na većinskom glasanju svih stabala

Slika 2.3: Algoritam metode slučajna šuma

2.3.5 Rotirajuća šuma

Rotirajuća šuma (eng. rotation forest) je također grupni klasifikacijski algoritam koji sadrži veći broj stabala odlučivanja, a predstavljen je 2006. godine [125]. Algoritam uključuje predobradu podataka koristeći *metodu glavnih komponenata* (eng. principal component analysis, PCA). PCA se koristi za odabir atributa, odnosno metrika u podskupu za učenje stabala. Kao i u metodi slučajne šume, svako stablo dobije drugačiji skup za učenje, a konačna odluka temelji se na većinskom glasanju svih stabala. Algoritam metode rotirajuće šume prikazan je slikom 2.4. Uz proizvoljno odabrane podskupove za učenje označene kao S_i te ukupan broj podskupova K, algoritam uključuje sljedeće korake [125]:

(1) Metrike iz skupa za učenje X podijele se u K nepreklapajućih podskupova veličine $M = n/K$;

(2) Iz svakog podskupa S_i ukloni se 25% uzoraka uporabom *bootstrap* metode, kako bi PCA dala drugačije koeficijente za svako stablo;

(3) Primijeni se PCA na preostalih 75% uzoraka i izračunaju se koeficijenti $a_{i,j}$ za svaki i-ti podskup i j-tu metriku unutar podskupa;

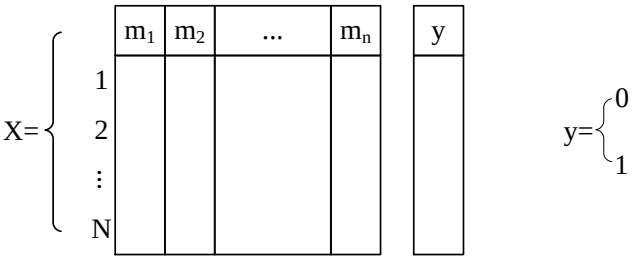
(4) Koeficijente $a_{i,j}$ se posloži u *rijetku* (eng. sparse) "rotirajuću" matricu R_k^a i presloži tako da pozicija koeficijenta odgovara rednom broju j-te metrike;

(5) Koraci (1) do (4) se ponavljaju za svako stablo koje se uči nad podskupom $X \cdot R_k^a, Y$;

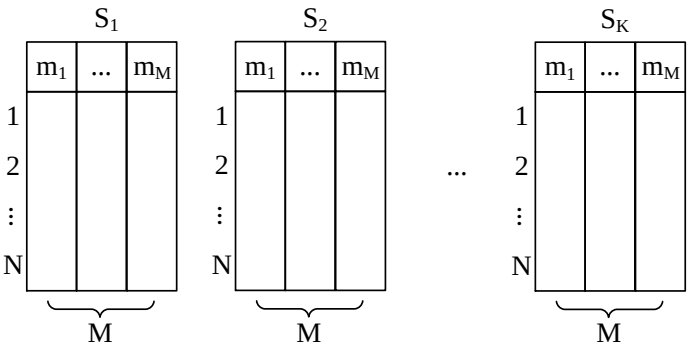
(6) Izlazna vrijednost grupne klasifikacije temelji se na većinskom glasanju svih stabala u šumi.

Bootstrap metodom dobivaju se različiti podskupovi podataka za učenje tako da se biraju nasumce, sa zamjenama, iz ukupnog skupa podataka za učenje.

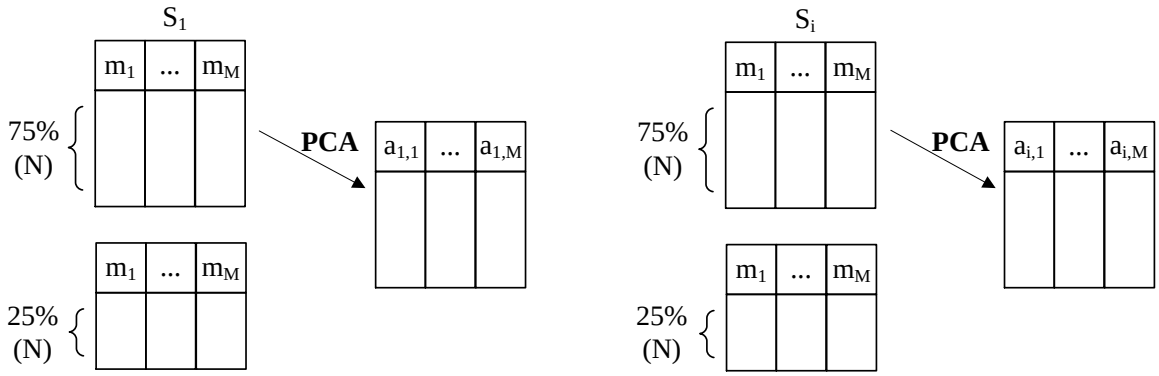
- Skup za učenje X sadrži *n* nezavisnih varijabli (metrika) i *y* kao zavisnu varijablu u *N* slučajeva



- Svaki nepreklapajući podskup S_i ($1 \leq i \leq K$) sadrži *M* slučajno odabranih metrika ($M = n / K$)



- Svakom podskupu S_i se nasumično ukloni 25% slučaja koristeći bootstrap metodu
- PCA računa koeficijente $a_{i,j}$ za svaku metriku ($1 \leq i \leq K, 1 \leq j \leq M$)



- Koeficijenti svih poskupova se unose u rijetku "rotirajuću" matricu R_k (k je indeks klasifikatora)

$$R_k = \begin{bmatrix} [\, a_{1,1}, \ldots, a_{1,M} \,] & \ldots & [\, 0 \,] & \ldots & [\, 0 \,] \\ & \ddots & & & \\ [\, 0 \,] & \ldots & [\, a_{i,1}, \ldots, a_{i,M} \,] & \ldots & [\, 0 \,] \\ & & & \ddots & \\ [\, 0 \,] & \ldots & [\, 0 \,] & \ldots & [\, a_{i,1}, \ldots, a_{i,M} \,] \end{bmatrix}$$

- Koeficijenti u R_k se preslaguju u $R_k{}^a k$ kako bi pozicija odgovarala rednom broju metrike
- Gradi se *k* stabala odlučivanja koristeći skup za učenje ($X \cdot R_k{}^a$, Y)
- Konačna klasifikacijska odluka se temelji na većinskom glasanju svih stabala

Slika 2.4: Algoritam metode rotirajuća šuma

Poglavlje 3

Proces prikupljanja podataka

Prvi korak u metodologiji ove disertacije je analiza trenutnog stanja u domeni predviđanja programskih neispravnosti, identifikacija praznina i definicija istraživačkih pitanja. Pritom je sustavni pregled literature najbolji način evaluacije i interpretacije dostupnog istraživačkog korpusa [30]. Sustavni pregled potreban je da bi se odgovorilo na sljedeća pitanja:

1. Koje su parametre koristile postojeće procedure prikupljanja podataka za SDP?
2. Koje su metrike programskih sustava sačinjavale reprezentativan skup podataka za SDP?
3. Koje su alate koristile postojeće procedure prikupljanja podataka za SDP?

Prvo pitanje motivirano je činjenicom da se prikupljanje podataka za SDP provodilo bez sustavnih smjernica kojima bi se naglasile sve nedvosmislene odluke koje su potrebne. Drugo pitanje teži pronalasku sveobuhvatnog skupa podataka, koji bi sadržavao što veći broj postojećih metrika programskog sustava. Treće pitanje motivirano je činjenicom da ljudski faktor može uzrokovati značajna odstupanja te ga kao takvog treba u što većoj mjeri zaobići korištenjem kvalitetnih alata sa što većim stupnjem automatizacije. Sva tri pitanja zajedno pomažu u definiciji **sustavnih smjernica za prikupljanje podataka za SDP**.

Drugi korak u metodologiji ove disertacije je empirijska istraživačka studija u kojoj se planira provedba ručnog prikupljanja podataka. Ručna analiza je primjeren način validacije ispravnosti i cjelovitosti procesa prikupljanja podataka za SDP [93]. Cilj ručne provedbe je sustavno definirati cjelokupni proces prikupljanja podataka koji bi poslužio u izradi alata za automatizirano prikupljanje. Istraživačka studija pritom ima sljedeće zadatke:

1. Otkriti parametre procedure koji nisu otkriveni sustavnim pregledom literature;
2. Analizirati utjecaj različitih vrijednosti parametara prikupljanja podataka;
3. Testirati alate za izračun metrika programskog sustava.

Prvi zadatak je potaknut činjenicom da različite studije navode različite parametre prilikom opisa procesa prikupljanja podataka. To ostavlja sumnju u valjanost konstrukcije provedenih istraživanja. Drugi zadatak teži kvantifikaciji utjecaja odabira vrijednosti parametara prikupljanja podataka na same podatke. Budući da parametri prikupljanja podataka mogu poprimiti više različitih vrijednosti, želi se provjeriti koliko to može izmijeniti završni skup podataka. Treći je zadatak pronaći odgovarajuće alate za izračun metrika programskog sustava (skraćeno, *alat za metrike*) koji bi se mogao integrirati u alat za automatizirano prikupljanje podataka.

3.1 Sustavni pregled literature

Sustavni pregled literature ove disertacije nadovezuje se na jednako sustavan pregled koji su na temu SDP-a proveli T. Hall i grupa autora [39]. Navedeni sustavni pregled literature proveden je uz jasno definirane kriterije za sva istraživanja koja su objavljena u razdoblju od 2000. do 2010. godine. Ova disertacija proširuje njihovu proceduru odabira znanstvenoistraživačkih radova na razdoblje od 2011. do 2014. godine.

Protokol i provedba sustavnog pregleda literature

Sukladno preporukama za provedbu sustavnog pregleda literature, nakon utvrđivanja potrebe potrebno je definirati protokol njegove provedbe [30]. Protokol je proveden u sustavnom pregledu literature za razdoblje od 2000. do 2010. godine pa je njegova kvaliteta već utvrđena [39]. Pretraga se provodi u dvije najčešće korištene baze znanstveno-istraživačkih radova u području programskog inženjerstva: IEEE Xplore i ACM Digital Library. Kriterij pretrage radova je izraz koji sadrži sljedeće ključne riječi: *(fault, bug, defect, error, correction, corrective, ili fix)* unutar naslova i *software* bilo gdje u tekstu. Kriterij pretrage radova također uključuje samo radove iz područja računarstva. Kriterij za odbacivanje radova u prvom ciklusu jest ukoliko se na temelju naslova i sažetka ustanovi da rad nije povezan s tematikom SDP-a. Drugi ciklus izbacivanja radova provodi se na temelju čitanja cjelovitog sadržaja prethodno prihvaćenih radova. Kao što je to učinjeno u sustavnom pregledu literature do 2010. godine [39], odbacuju se radovi koji nemaju opis prikupljanja podataka za SDP. Radovi koji zadovolje sva tri kriterija (ključne riječi i područje istraživanja, SDP tematika i opis prikupljanja podataka) pridodani su popisu radova iz sustavnog pregleda literature do 2010. godine [39]. Tablica 3.1 predstavlja broj radova koji su bili prisutni u svakom od koraka procedure njihova odabira za pregled. Re-

Tablica 3.1: Proces odabira radova

Opis koraka	IEEE Xplore	ACM Digital Library	Ukupno
Radovi preuzeti temeljem ključnih riječi	2000	2447	4447
Izbacivanje temeljem područja, naslova i sažetka	-1935	-2376	-4311
Radovi uzeti u obzir za pregled	65	71	136
Izbacivanje temeljem cjelokupnog sadržaja	-50	-51	-101
Radovi prihvaćeni za pregled	15	20	35

zultat je 136 detaljno pregledanih radova iz razdoblja 2011. - 2014. godine, od kojih 35 radova zadovoljava sve kriterije, te još 36 radova koji su preuzeti iz sustavnog pregleda literature do 2010. godine. Valja napomenuti da među odbačenim radovima može postojati opis prikupljanja podataka, ali takav da iz nekog razloga nije zahtijevao povezivanje razvojnih repozitorija. Neki radovi proučavaju samo neispravnosti i njihove karakteristike te nemaju potrebu za analizom programskog koda. Među odbačenim radovima bili su i oni koji koriste gotove skupove podataka koji nemaju opis prikupljanja, poput onih iz repozitorija PROMISE (eng. Prediction Models In Software Engineering) [137].

Protokol prikupljanja korisnih informacija građen je iterativno, nadopunjujući popis parametara prikupljanja podataka sa svakom novom studijom. Sažimanje podataka temeljeno je na opisnoj analizi koja tablično predstavlja otkrivene parametre prikupljanja podataka, njihove sličnosti i njihove različite vrijednosti.

3.1.1 Parametri prikupljanja podataka otkriveni u literaturi

Analiza odabranih radova i usporedba opisanih procedura za prikupljanje podataka za SDP ukazale su na nekoliko parametara po kojima se razlikuju. U nastavku ovoga poglavlja, predstavit će se parametri prikupljanja podataka te njihove vrijednosti koje su mogle uzrokovati odstupanja postojećih procedura. Sažetak parametara prikupljanja podataka i korištenih vrijednosti prikazan je u tablici 3.2.

Razina granulacije programskog sustava

U analizi objektno orijentiranog programskog koda postoji nekoliko mogućih razina granulacije. Programske sustave moguće je promatrati na razini metoda, razreda, datoteka, paketa ili pak cjelokupnog sustava. Odabir razine granulacije direktno utječe na raspon metrika jer

su pojedine metrike programskog sustava svojstvene samo određenoj razini granulacije, te na distribuciju neispravnosti unutar modula programskog sustava. Taj odabir, posljedično, može utjecati i na rezultate istraživanja jer određena ponašanja ne moraju biti prisutna na svim razinama.

Promatranje programskog sustava na **razini sustava** nema pretjerano smisla za SDP jer je njegov cilj usmjeriti testiranje na potencijalno problematične programske module. Preostale, prethodno navedene razine mogu biti valjane. Svaka niža razina granulacije daje veći broj modula programskog sustava za analizirati te predviđanje postaje određenije. **Razina paketa** je najrjeđe korištena. Glavni je razlog tomu to što se, prema potrebi, ona jednostavno može izvesti iz razine datoteka, te da su moduli programskog sustava tada značajnije veći [45]. Pojedini radovi koristili su varijantu paketa, svrstavajući datoteke u grupe od po četiri [138]. **Razina metoda** također je korištena rijetko u drugim radovima, ali i u gotovim skupovima podataka poput onih iz PROMISE* repozitorija. **Razina datoteka** najčešće je korištena i u drugim radovima i u gotovim skupovima podataka iz Eclipse Bug Data† repozitorija. **Razina razreda** je druga najčešće korištena razina, koja je, pored radova iz tablice 3.2, prisutna i u gotovim skupovima podataka iz repozitorija PROMISE i *Bug prediction dataset*‡.

Budući da je sustav za upravljanje inačicama temeljen na razini datoteka, ona se smatra dobrim izborom za razinu granulacije [45]. Razlika između razine datoteka i razreda ponekad je vrlo malena, poput slučaja kada se analiziraju samo *javni razredi* (eng. public class). U tom slučaju, studije poput [45] navode nemogućnost razlikovanja javnih i ugniježđenih razreda kao *prijetnju valjanosti istraživanja* (eng. threat to validity). S druge strane, neke studije zanemaruju taj problem te proglašavaju razinu razreda, izjednačavajući datoteke i javne razrede [139], ili pak ne spominju razinu datoteka [140]. Pored toga, važno je biti svjestan i datoteka i razreda koji nisu relevantni za ponašanje programskog sustava i SDP, poput testova i primjera [45].

Metrike programskog sustava

Odabir razine granulacije ograničava raspon metrika koje je moguće izračunati iz modula programskog sustava. Nije neuobičajeno da se koriste neke metrike važne za razumijevanje ponašanja sustava, iako su izračunate na nižoj razini. Uobičajeni način za podizanje metrike na višu razinu podrazumijeva deskriptivnu analizu vrijednosti te metrike za sve module programskog

*http://openscience.us/repo
†http://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/
‡http://bug.inf.usi.ch/

sustava koji pripadaju modulu više razine. Računa se minimalna, maksimalna i srednja vrijednost te medijan i suma svih vrijednosti neke metrike za sve module niže razine i pripisuju se modulu više razine. Zamjerka tog postupka je da se podaci svakom transformacijom udaljavaju od originalnog stanja te da su zaključci dobiveni nad takvim podacima stoga slabiji [26].

Metrike programskog sustava korištene za SDP mogu se svrstati u tri kategorije [113]:

1. *Metrike proizvoda* (eng. product metrics);
2. *Metrike razvojnog procesa* (eng. development process metrics);
3. *Metrike implementacije i uporabe* (eng. deployment and usage metrics).

Metrike proizvoda su najčešće korištene metrike. Sastoje se od raznih statičkih metrika veličine i složenosti, objektno orijentiranih metrika ili *metrika napora* (eng. effort metrics) koje se mogu izračunati iz programskog koda. Navedene metrike prisutne su u gotovim skupovima podataka iz repozitorija PROMISE i *Bug prediction dataset*. Dominantna uporaba tih metrika potvrđena je još jednim sustavnim pregledom literature koji je analizirao metrike za SDP [141]. **Metrike razvojnog procesa** se u pojedinim radovima koriste kao dodatak metrikama proizvoda. One se uobičajeno računaju iz promjena u programskom kodu, poput broja izmijenjenih, izbrisanih ili dodanih linija koda, broja ljudi koji sudjeluju u izmjenama, broja poteškoća na koje su naišli ili vremenskog aspekta njihova djelovanja [71, 72, 138, 142]. **Metrike implementacije i uporabe** rijetko se susreću u analiziranim radovima, a nalaze se u rijetko korištenim gotovim skupovima podataka koji se tiču samo *izvještaja neispravnosti* (eng. bug report datasets) unutar repozitorija PROMISE. Te metrike ponajprije se odnose na starost i detalje izdavanja proizvoda, poput broja inačice, vremena koje je proteklo prije ili poslije izdanja određene inačice ili vremena izvršavanja programskog koda [138, 142, 143].

Jedan od razloga zašto su metrike proizvoda i metrike razvojnog procesa dominantno prisutne u srodnim istraživanjima jest taj da se one mogu prikupljati relativno brzo i jednostavno, čak i u velikim projektima [144].

Težina neispravnosti

Težina neispravnosti se, sukladno normi IEEE 1044-2009 [10] i opisu iz poglavlja 2.1, može svrstati u nekoliko kategorija prilikom njezina unosa u repozitorij sustava za praćenje zahtjeva. Kategorija trivijalnih neispravnosti odnosi se na jednostavne tipografske pogreške uslijed kojih ne dolazi do pogreške u radu sustava. Također, postoji kategorija zahtjeva koji ne predstavlja neispravnost, već je zahtjev za *unapređenjem* (eng. enhancement). Navedene dvije kategorije

ne podrazumijevaju gubitak funkcionalnosti, nisu značajne za otkrivanje kvarova, te kao takve nisu od interesa za SDP [140, 145]. Neke studije propustile su jasno definirati koje kategorije težine neispravnosti uzimaju u obzir. Primjerice, Concas i grupa istraživača [146] istraživali su mogućnost korištenja metrika koje potječu iz društvenih mreža za SDP u objektno orijentiranim sustavima Eclipse i Netbeans. Pridržavaju se upute da je ispravna razina težine neispravnosti ona koja ne uključuje zahtjeve za unapređenjem [147], ali ne komentiraju kategoriju trivijalnih neispravnosti. Postoje radovi koji jasno ističu da koriste samo *ispravljene* (eng. fixed) neispravnosti koje spadaju u kategoriju blokirajuće, kritične, velike i normalne težine iz repozitorija sustava Bugzilla [73, 140]. Međutim, ne komentiraju zašto ne koriste i kategoriju *malih* (eng. minor) neispravnosti, koja također podrazumijeva gubitak funkcionalnosti.

Sustavi za praćenje zahtjeva, pored kategorije težine sadrže i druge važne kategorije koje opisuju njihove unose. **Status neispravnosti** je važan jer razlikuje otvorene (nepotvrđene, potvrđene i one na kojima se trenutno radi) i zatvorene neispravnosti (riješene i verificirane). **Rezolucija zatvorenih neispravnosti** može biti kategorizirana kao *ispravljena* (eng. fixed), *nevažeća* (eng. invalid), *neispravljiva* (eng. wont fix), *duplicirana* (eng. duplicated) ili *neponovljiva* (eng. works for me). Iako je evidentno da samo ispravljene neispravnosti možemo pronaći u repozitoriju sustava za upravljanje inačicama, važno je to i naglasiti u opisu procesa prikupljanja podataka, kao što je učinjeno u nekim radovima [73, 140]. Ukoliko bi se iz nekog razloga koristile druge kategorije statusa ili rezolucije, to bi moglo uzrokovati velika odstupanja [33].

Redoslijed pretrage repozitorija

Broj neispravnosti po modulu programskog sustava sastavni je dio skupova podataka za SDP, a dobiva se povezivanjem podataka iz razvojnih repozitorija sustava za praćenje zahtjeva i sustava za upravljanje inačicama. Povezivanje podataka moguće je započeti tako da se prikupe **prvo neispravnosti** ili **prvo predaje**. Pristup povezivanja podataka koji prikuplja prvo neispravnosti traži predaju izmjene programskog koda koja u svojemu opisu sadrži poveznicu na neku neispravnost [73, 140]. To je najčešće identifikacijska oznaka (ID) neispravnosti, ali može biti vrijeme nastanka, autor ili neki detalj iz opisa ili komentara neispravnosti. Popularni **pristup SZZ** prikuplja prvo predaje te svaku brojku unutar njihovih opisa smatra kao potencijalnu neispravnost [68]. Postoji primjer studije koja je, prikupljajući prvo predaje, uspješno povezala 37% neispravnost za ArgoUML, odnosno 43% neispravnosti za Eclipse BIRT projekt [73]. Postoji i

studija koja je uspješno povezala 47% neispravnosti za nekoliko inačica Mozilla projekta, prikupljajući prvo neispravnosti [84]. Budući da dvije navedene studije nisu provele prikupljanje iz istog izvora, teško je usporediti uspješnost dvaju pristupa. Većina radova niti ne komentira odabir pristupa. Međutim, jedna studija je to učinila i demonstrirala njihovu različitu uspješnost koristeći isti izvor podataka. Podaci su prikupljani iz 52 *dodatka* (eng. plug-in) iz razvojne zajednice Eclipse. Povezano je 50% neispravnosti prikupljajući prvo predaje, a dodatnih 20% prikupljajući prvo neispravnosti. Iz toga se može zaključiti da ovaj parametar prikupljanja podataka može uvelike utjecati na točnost podataka i valjanost konstrukcije istraživanja.

Tablica 3.2: Parametri prikupljanja podataka u postojećim procedurama

Parametar	Mogućnosti	Reference
Razina granulacije	Sustav	-
	Paket	[138], [148], [149]
	Datoteka	[140], [78], [79], [80], [71], [150], [151], [152], [153], [154], [155], [156], [157], [158], [3], [23], [77], [149], [106], [159], [160], [161]
	Razred	[45], [84], [140], [73], [162], [139], [149], [163]
	Metoda	[73], [164], [165], [166]
Metrike	**Proizvoda**	[3], [45], [84], [138], [140], [71], [142], [167], [145], [139], [148], [79], [156], [158], [65], [77], [159], [160], [166], [168], [75]
	Razvojnog procesa	[71], [142], [150], [138], [45], [139], [148], [156], [77] [169], [163], [106], [159], [166], [75], [161]
	Implementacije i uporabe	[142], [138]
Težina neispravnosti	**Iznad trivijalnih**	[73], [140], [145]
	Uključuje trivijalne	[146], [157]
	Uključuje unapređenja	[154]
Redoslijed pretrage repozitorija	Prvo predaje izmjena	[92], [106], [71], [142], [65], [152], [154], [157], [77], [159], [160], [166], [169], [170], [171], [69], [90]
	Prvo neispravnosti	[170], [155], [69], [158], [165], [149]

3.2 Istraživačka studija

Prva analiza slučaja u ovoj disertaciji za svrhu ima istraživanje utjecaja parametara prikupljanja podataka na dobivene podatke. Budući da je to i jedina analiza slučaja koja ima za cilj isključivo istraživanje, u nastavku disertacije oslovljava se kao istraživačka studija.

Metodologija istraživačke studije

Provedba istraživačke studije napravljena je sukladno preporukama za provedbu analize slučaja [28]. Protokol studije zasnovan je na ručnoj provedbi procesa prikupljanja podataka. Višestrukost izvora podataka osigurana je uporabom nekoliko projekata otvorenoga koda: JDT, PDE, Platform, BIRT i Mylyn. Odabrani projekti pripadaju razvojnoj zajednici Eclipse, koja je ujedno i najčešće korištena u srodnim studijama. Ti projekti imaju najduže trajanje (razvoj traje preko jednog desetljeća) i najveći broj prijavljenih neispravnosti. Njihovi repozitoriji iz sustava za upravljanje inačicama [§] i sustava za praćenje zahtjeva[¶] javno su dostupni.

Materijal ove istraživačke studije čine: repozitoriji sustava za upravljanje inačicama i sustava za praćenje zahtjeva, alati za metrike i forme za prikupljanje podataka. Razvojna zajednica Eclipse koristi repozitorij za upravljanje inačicama sustava GIT i repozitorij za praćenje zahtjeva, odnosno neispravnosti sustava Bugzilla. Alati za metrike detektirani su u srodnim radovima i pretragom na Internetu. Forme za prikupljanje podataka pripremljene su i detaljno obrazložene u zadacima koji su dani sudionicima ove istraživačke studije:

Forma 1 prikuplja sljedeće informacije iz repozitorija neispravnosti: ID neispravnosti, ime i inačice produkta, cjelovit sažetak i prioritet neispravnosti, osoba kojoj je neispravnost dodijeljena, broj komentara, datum prijave i posljednje izmjene statusa;

Forma 2 dokumentira poveznice neispravnosti i predaja izmjena programskog koda kroz: ID neispravnosti, povezan (da/ne) i ID predaje;

Forma 3 pohranjuje podatke o datotekama izmijenjenim u povezanim predajama: ID neispravnosti, putanja datoteke, broj unesenih i izbrisanih linija koda;

Forma 4 zapisuje metrike povezanih datoteka koje se računa nekim od alata za metrike: putanja datoteke, metrike ($m1$, $m2$, ... m_n);

Forma 5 predstavlja izgled konačnog skupa podataka za SDP: ime i inačica programskog sustava, putanja datoteke, metrike ($m1$, $m2$, ... m_n), broj neispravnosti.

[§] https://git.eclipse.org/

[¶] https://bugs.eclipse.org/

Sudionici koji provode istraživačku studiju su 12 studenata poslijediplomskog studija računarstva s Tehničkog fakulteta u Rijeci. Studenti su polaznici kolegija Upravljanje u programskom inženjerstvu u kojem se uče važnosti ispravnih i cjelovitih podataka koji pomažu procesima donošenja odluka u razvoju velikih i složenih programskih sustava. Sudionici imaju raznovrsnu razinu prethodnog iskustva u uporabi repozitorija sustava za upravljanje inačicama, ograničeno iskustvo s repozitorijima za praćenje zahtjeva, te nemaju iskustva sa alatima za metrike. Stoga se svi sudionici upoznaju sa potrebnim konceptima istraživačke studije kroz pripremne zadatke i seminare.

Zadaci kroz koje sudionici provode istraživačku studiju su sljedeći:

1. Ispitati prikladnost alata za izračun metrika programskog koda;
2. Provesti prikupljanje podataka ručno i automatizirano;
3. Analizirati prikupljene podatke i kvantificirati odstupanja.

Prvi zadatak mora biti proveden prije prikupljanja podataka jer o njegovom ishodu ovisi koji će se alati detaljnije analizirati kroz uporabu. Kombinacija ručnog i automatiziranog prikupljanja podataka pomaže umanjiti odstupanje koje uvodi sudionik i otkriti skrivene parametre prikupljanja podataka koje također treba definirati.

Metode koje se koriste u provedbi zadataka su:

1. Seminari u kojima se sudionici upoznaju s temom istraživanja i alatima za izračun metrika programskog koda;
2. Pismene instrukcije potkrijepljene primjerima za svaki korak u prikupljanju podataka;
3. Upute za analizu i validaciju rezultata prikupljanja podataka.

Nakon provedene pretrage za alatima, sudionicima su dodijeljeni seminari u kojima se ispituje prikladnost alata. Kroz te seminare, sudionici se upoznavaju s alatima i problemima prikupljanja te se provodi analiza uporabljivosti u konkretnoj primjeni.

Dizajn istraživačke studije pripremljen je tako da minimizira sva odstupanja u prikupljanju podataka na koja može utjecati. Sudionicima su dodijeljeni personalizirani zadaci kroz 4 vježbe s ukupno 52 koraka detaljno opisana i potkrijepljena 21 slikom preslike ekrana. Kako sami sudionici mogu uzrokovati odstupanja, primijenjen je princip *triangulacije promatrača* (eng. observer triangulation) [92]. Vremenski rokovi provedbe zadataka definirani su tako da osiguraju dovoljno vremena za analizu rada alata za metrike i eventualne korektivne mjere.

3.2.1 Parametri prikupljanja podataka otkriveni u primjeni

Neki parametri prikupljanja podataka otkriveni su u literaturi i opisani u poglavlju 3.1.1 te sažeti u tablici 3.2. Istraživačka studija ima za cilj otkriti postoje li dodatni parametri kojima nije posvećena pažnja u srodnoj literaturi. U nastavku ovoga poglavlja predstavit će se parametri prikupljanja podataka koji su otkriveni u primjeni, te njihove vrijednosti koje mogu uzrokovati odstupanja u prikupljenim podacima.

Odabir alata za metrike

Pretraga srodne literature i Interneta pronašla je 35 alata za metrike. Alati su analizirani u tri faze te su u svakoj uspoređivani po određenim kriterijima. Da bi se alati za metrike mogli upotrijebiti u alatu za automatizirano prikupljanje, trebali su biti sposobni analizirati velike Eclipse projekte pisane u programskom jeziku Java. Reprezentativni skup metrika koje se pri tome trebaju računati sačinjavaju metrike produkta na razini datoteka ili razreda, a izlazno izvješće trebalo bi biti generirano u formatu koji se lako može raščlaniti. **Parametar 1.1.** odnosi se na alat za metrike koji treba zadovoljiti sljedeće kriterije:

1. **Dostupnost alata** - za testiranje u primjeni;
2. **Podrška alata** - za programski jezik Java;
3. **Razina granulacije** - za razinu datoteka ili razreda;
4. **Metrike** - produkta;
5. **Ulazna uporabljivost** - sposoban za prihvat velikih projekata;
6. **Izlazna uporabljivost** - generiranje izvješća u csv, html ili xml formatu.

Tablica 3.3 prikazuje koliko alata je odbačeno po svakom od kriterija. Pored kriterija predviđenih parametrom 1.1., neki alati bili su odbačeni jer su predstavljali stariju inačicu nekog drugog alata, njihove probne inačice nisu pružale mogućnost kompletnog testiranja ili čak nije niti bila riječ o alatu nego o platformi za drugačiju namjenu. Po kriteriju kojim se zahtijevala mogućnost izračuna metrika produkta izbačeno je čak 8 alata. Od toga, 4 alata nisu pružala tu mogućnost, a druga 4 su imala premali broj metrika koje računaju. Ti alati računali su svega nekoliko metrika koje su pokrivene svim ostalim alatima koji su zadovoljili taj kriterij s mnogo većim brojem metrika.

Potpuni popis alata za metrike koji su analizirani je sljedeći: iPLASMA, inFusion, inCode, CodeCover, JavaNCSS, ckjm, Understand, Borland Together, CodeSonar Static Analysis Tool, CodePRO Analytix, Metrics 1.3.8, Moose, JavaMetrics, Testwell CMT ++ (CMTJava), JDe-

pend, Dependency Finder, JarAnalizer, CCCC, Source Code Metrics, Classycle, Sonar, Resource Standard Metrics, Jhawk, Jtest, SonarGraph, PMD, JDiff, CodeCount, LocMetrics, CodeAnalyzer, JMT, Jmetric, ES2, Xradar, Essential Metrics. Prva faza svodila se na analizu opisa alata, a u njoj je odbačeno 16 od 35 alata. U drugoj fazi se alate detaljnije analiziralo kroz seminare i primjenu nad primjerom projekta te je u njoj odbačeno 14 od 19 alata. U trećoj fazi se alate analiziralo kroz ručnu i automatiziranu primjenu u prikupljanju podataka. Analiziralo se sljedeće alate: CodePRO Analytix, JHawk, LOC Metrics, Metrics 1.3.8 i Source Code Metrics. Na kraju su odabrana 2 alata koja udovoljavaju svim kriterijima: JHawk i LOC Metrics. Metrics 1.3.8 je zahtijevao izgradnju kompletnih projekata, što je zbog njihove veličine znatno usporavalo proces. Source Code Metrics je imao grešku uslijed koje bi svaka iduća analiza zahtijevala ponovno pokretanje NetBeans razvojne okoline. Budući da treba omogućiti prikupljanje podataka iz velikog broja inačica projekata, ta dva alata nisu prihvaćena za konačnu primjenu. Code PRO Analytix je imao ograničenje po veličini projekata i nije zadovoljavao kriterij 5. Kako nije pružao mogućnost automatiziranog pokretanja, i on je bio odbačen. LOC Metrics se pokazao kao jednostavan i iznimno brz besplatni alata koji računa 10 jednostavnih metrika programskog sustava. JHawk je komercijalni alat koji nadopunjuje LOC Metrics s čak 40 metrika produkta, te tako zajedno sačinjavaju najširi popis metrika u cjelokupnoj srodnoj literaturi. Potpuni popis metrika i njihovi opisi nalaze se u prilogu P1. Oba alata pružala su mogućnost automatizacije te su prihvaćena za konačnu primjenu.

Tablica 3.3: *Parametar 1.1.* - analiza alata za metrike

#	Kriterij	Opis	Rezultat
1	**Dostupnost alata**	besplatno ili uz probnu inačicu za testiranje u primjeni	6 odbačena
2	**Podrška alata**	za programski jezik Java	0 odbačena
3	**Razina granulacije**	za razinu datoteka ili razreda	7 odbačena
4	**Metrike**	produkta koje se računaju iz programskog koda	4 + 4 odbačena
5	**Ulazna uporabljivost**	sposoban za prihvat velikih projekata	5 odbačena
6	**Izlazna uporabljivost**	generiranje izvješća u csv, html ili xml formatu	1 odbačen
7	**Nepredviđeni**	stara ili neodgovarajuća inačica alata	2 odbačena
		nije alat nego platforma	2 odbačena
		probna inačica je previše limitirana	2 odbačena

Odabir projekata otvorenog koda

Prije početka prikupljanja podataka, potrebno je odabrati projekte koji će biti izvor podataka za istraživanje u području SDP. Iako se čini trivijalnim, prikupljanje neće biti moguće ukoliko se neki podaci ne podudaraju. **Parametar 2.1.** zahtijeva da odabrani projekt posjeduje repozitorije sustava za praćenje neispravnosti i sustava za upravljanje inačicama. **Parametar 2.2.** zahtijeva povezivost oznaka za inačicu projekata unutar repozitorija oba sustava.

Za projekte koji nemaju oba razvojna repozitorija ili nemaju usklađene oznake inačica, nije moguće provesti povezivanje neispravnosti i programskog koda. Ime i *etikete inačice* (eng. release tag) moraju se podudarati unutar oba repozitorija kako bi se moglo provesti povezivanje neispravnosti i programskog koda. Procjena važnosti tih parametara provedena je tako da se za velik broj projekata analiziralo posjeduju li odgovarajuće oznake. Odabrano je 85 projekata iz razvojne zajednice Eclipse koji su bili sadržani u njihovom repozitoriju Bugzilla sustava za praćenje neispravnosti. Tablica 3.4 prikazuje da je 76 od 85 projekata bilo pohranjeno unutar njihovog repozitorija GIT sustava za upravljanje inačicama. Za projekte koji su bili u oba repozitorija, kod 51 od 76 projekata oznake inačica su bile povezive. To dokazuje da neki projekti nemaju usvojeno pravilo imenovanja inačica, što može onemogućiti prikupljanje podataka za SDP.

Tablica 3.4: *Parametri 2.1. i 2.2.* - odabir projekata

Zahtjev	Karakteristika	Broj projekata
-	Ukupan broj analiziranih projekata:	85
2.1.	Posjeduju oba razvojna repozitorija	76
2.2.	Posjeduju odgovarajuće oznake inačica projekata	51

Prikupljanje iz repozitorija neispravnosti

Kako je bilo spomenuto u potpoglavlju 3.1.1, koje je govorilo o redoslijedu pretrage repozitorija, zaključeno je da je uspješnost povezivanja podataka veća kada prikupljanje započne od repozitorija sustava za praćenje neispravnosti. Parametri prikupljanja podataka u literaturi su otkrili i kategorije neispravnosti koje su relevantne za SDP. **Parametar 3.1.** zahtijeva da težina neispravnosti bude relevantna za SDP. **Parametar 3.2.** zahtijeva da status neispravnosti bude "pozitivno razriješena".

Unutar sustava za praćenje zahtjeva važno je razlikovati neispravnosti od zahtjeva za unapređenjem. Također, važno je razlikovati neispravnosti trivijalne težine, koje ne podrazumijevaju gubitak funkcionalnosti, od onih koje su teže i relevantne za SDP. Budući da se neispravnosti moraju moći povezati s programskim kodom, moraju imati zatvoreni status i rezoluciju koja upućuje da su ispravljene. Forma 1 se zato popunjava samo neispravnostima koje zadovoljavaju odabrane parametre:

- **Status:** *razriješen* (eng. resolved), verificiran ili zatvoren;
- **Rezolucija:** ispravljena;
- **Težina:** blokirajuća, kritična, velika, normalna ili minorna.

Koristeći navedene parametre, zaobilazi se sve nepotvrđene, netočne ili duplicirane unose, neispravnosti koje još nisu ispravljene ili koje se nisu mogle ispraviti, te zahtjeve za unapređenjem. Provedena je procjena mogućeg odstupanja u prikupljanju podataka ako bi se ti parametri ostavili otvorenima za interpretaciju. Uspoređeni su broj neispravnosti koje zadovoljavaju navedene parametre i broj neispravnosti suprotnih parametara. Suprotni parametri za status su: nepotvrđeni, novi, dodijeljeni i ponovno otvoreni, za rezoluciju su: nevažeća, neispravljiva, duplicirana i neponovljiva, a za težinu su: trivijalna i zahtjevi za unapređenjem.

Tablica 3.5: *Parametri 3.1. i 3.2.* - odabir parametara neispravnosti

Zahtijevani parametri	**Broj neispravnosti po projektima**				
	JDT	PDE	Platform	BIRT	Mylyn
Odabrani parametri	20353	7318	38511	13071	3492
Suprotan status	0	0	0	0	0
Suprotna rezolucija	16488	3570	32582	3059	1909
Suprotna težina	2875	895	4604	1243	1581
Nepotpuni podaci	64	26	108	7	1

Tablica 3.5 prikazuje rezultat navedene analize. Pored suprotnih parametara, uočeno je da u repozitoriju sustava za praćenje zahtjeva postoje i nepotpuni unosi. Takvim unosima nedostaje neki od podataka ključnih za povezivanje poput identifikacijske oznake, imena ili inačice projekta. Rezultati ukazuju da je vrlo važno odabrati odgovarajuće parametre neispravnosti te da možemo očekivati velike razlike u prikupljenim podacima ukoliko neke od njih izmijenimo. To prije svega vrijedi za odabir rezolucije koji može izmijeniti broj neispravnosti od 20% do čak 45% ukoliko bi ih se sve uključilo u prikupljanje podataka. Neispravnosti suprotnog statusa ne

predstavljaju problem jer su ograničene odabirom rezolucije, odnosno, za odabranu rezoluciju (ispravljena) ne postoje neispravnosti suprotnih parametara. U većini projekata, neispravnosti suprotne težine mogu uzrokovati odstupanje od 10% do 15% ukoliko bi ih se uključilo u prikupljanje podataka. Za Mylyn projekt mogu uzrokovati odstupanje od čak 32%. Nepotpuni podaci ne predstavljaju značajna odstupanja, ali narušavaju kvalitetu podataka jer se ne mogu povezati s programskim kodom. Odabir odgovarajućih parametara neispravnosti važan je jer može utjecati na valjanost konstrukcije istraživanja.

Prikupljanje iz repozitorija sustava za upravljanje inačicama

Druga faza prikupljanja podataka podrazumijeva pronalazak stabilnih inačica projekata unutar repozitorija sustava za upravljanje inačicama. Stabilna inačica je ona koja je izdana široj korisničkoj zajednici. U razvojnoj zajednici Eclipse, takve inačice se označavaju dvjema znamenkama (npr. 2.0, 2.1) i izdaju se nakon unosa novih funkcionalnosti. One su nam potrebne kako bismo izračunali vrijednosti metrika za programski kod koji je sadržavao određene neispravnosti. Neispravnosti koje se vežu na takvu inačicu su one koje se pronađu nakon njezinog izdavanja. Potrebno je razlikovati stabilne inačice od međuinačica, koje se izdaju nakon ispravljanja određenog broja neispravnosti. Međuinačice se u zajednici označavaju Eclipse sa trima znamenkama (npr. 2.0.1, 2.0.2, 2.0.3). Ukoliko su zadovoljeni zahtjevi iz parametara 2.1. i 2.2., za velike projekte moguće je da su podijeljeni u više komponenata i da svaka od njih ima svoj podrepozitorij. **Parametar 4.1.** zahtijeva prikupljanje svih podrepozitorija za odabrani projekt unutar glavnog repozitorija razvojne zajednice.

Tablica 3.6: *Parametar 4.1.* - pronalazak podrepozitorija

Zahtjev	Karakteristika	Rezultat
-	Ukupan broj analiziranih projekata:	85
4.1.	>1 podrepozitorij za upravljanje inačicama	37

Tablica 3.6 prikazuje za koliko projekata je pronađeno više od jednog podrepozitorija. Od 85 analiziranih projekata, čak 37 ih je raspodijeljeno u više od jednog podrepozitorija. Ukoliko bi se navedeni parametar zanemario, cjelokupnost podataka mogla bi biti vrlo upitna.

Povezivanje neispravnosti i predaja

Najvažnija i najzahtjevnija faza u prikupljanju podataka za SDP je povezivanje repozitorija neispravnosti i programskog koda. Ishod ove faze je broj neispravnosti za module programskog sustava, što predstavlja zavisnu varijablu za SDP. Svaka izmjena programskog koda pohranjuje se u formatu predaja u sustav za upravljanje inačicama. Najčešća i najvjerodostojnija tehnika povezivanja neispravnosti i predaja je potraga ID-a neispravnosti unutar opisa predaja.Tom su tehnikom prikupljani podaci unutar ove istraživačke studije, a pritom su se pohranjivali unutar Forme 2. Ručnom analizom prikupljenih podataka otkriveni su novi parametri. **Parametar 5.1.** ukazuje da se sam broj ID-a neispravnosti može nalaziti u sklopu većeg bloka znamenaka. **Parametar 5.2.** ukazuje da se ID neispravnosti može nalaziti u opisu većeg broja predaja.

ID oznaka neispravnosti može biti različite duljine. Pored toga, opisi predaje izmjena ponekad sadrže i druge identifikacijske oznake, datume ili druge brojčane vrijednosti. To predstavlja poteškoću u automatiziranom prikupljanu podataka. Kako bi se kvantificiralo odstupanje koje može unijeti pristup koji ne bi uzimao u obzir parametar 5.1., uspoređuje se *stopa povezivanja* (eng. linking rate) za jednostavnu pretragu i strogu pretragu. Stopa povezivanja izražava se postotkom neispravnosti koji se poveže s nekom predajom. Jednostavna pretraga opisana je u poglavlju 2.1.1. Ona predstavlja automatizirano povezivanje podataka koje samo traži broj ID neispravnosti unutar opisa. Stroga pretraga ručnom analizom odbacuje veze za koje se ustanovi da opis predaje ne sadrži ID neispravnosti, već neki drugi broj ili oznaku. Tablica 3.7 dokazuje važnost definiranja preciznog kriterija za povezivanje ID-a neispravnosti i opisa predaja izmjena koda. Odstupanje uzrokovano nedovoljno preciznim kriterijem može biti od 7% do čak 40%.

Tablica 3.7: *Parametar 5.1.* - pretraga ID-a neispravnosti u opisu predaja

Način pretrage	**Stopa povezivanja po projektima**				
	JDT	PDE	Platform	BIRT	Mylyn
Broj neispravnosti:	18696	6822	34641	8101	2739
Jednostavna pretraga:	80.5%	65.8%	80.7%	59.8%	52.3%
Stroga pretraga:	72.0%	58.6%	71.2%	18.9%	16.1%

Utjecaj parametra 5.2. kvantificira se usporedbom broja neispravnosti i broja predaja koje su povezane s neispravnostima nakon ručne pretrage. Tablica 3.8 dokazuje da veza neispravnosti i predaja nije kardinalnosti jedan-na-jedan ($1 : 1$) jer broj povezanih predaja premašuje broj povezanih neispravnosti. Ovaj parametar je važno imati prilikom izgradnje procedure za

automatizirano prikupljanje kako se ne bi izostavilo značajnu količinu informacija. Ukoliko bi se izostavilo parametar 5.2., valjanost konstrukcije istraživanja bila bi značajno narušena.

Tablica 3.8: *Parametar 5.2.* - veza neispravnosti i predaja

Broj	**Višestruke veze po projektima**				
povezanih	JDT	PDE	Platform	BIRT	Mylyn
Neispravnosti:	13468	3995	24663	1532	441
Predaja:	17667	7573	29669	2127	3002

Izgradnja konačnog skupa podataka

Odabrana razina granulacije modula programskog sustava je razina datoteka. Završna faza procedure prikupljanja podataka je izgradnja konačnog skupa podataka koji sadrži vrijednosti odabranih metrika programskog sustava i broj neispravnosti za svaku datoteku u inačici nekog projekta. Za tu namjenu koristi se Forma 5 u koju je potrebno povezati podatke dobivene povezivanjem neispravnosti i datoteka te korištenjem alata za metrike. Za povezivanje neispravnosti i datoteka koristi se Forma 3, koja uvodi uređene DN (Datoteka, Neispravnost) parove. Za analizu alatima za metrike koristi se Forma 4, koja posjeduje podatke o datotekama. **Parametar 6.1.** zahtijeva uporabu jedinstvene oznake svake datoteke unutar projekta kako bi se mogle povezati Forma 3 i 4. **Parametar 6.2.** zahtijeva da kardinalnost veze datoteka - neispravnosti bude više-na-više ($m : n$). **Parametar 6.3.** ukazuje na mogućnost dupliciranih veza između jedne datoteke i jedne neispravnosti.

Putanja datoteke nameće se kao logičan izbor identifikacije datoteka jer ne mogu postojati dvije identične putanje. Putanja datoteka se također nalazi i u izlaznim izvješćima alata za metrike, što olakšava automatizirano prikupljanje. Parametar 6.2. je posljedica načina rada programera te broja i složenosti neispravnosti koje se pronađu. Ispravljanje nekih neispravnosti zahtijeva izmjenu većeg broja datoteka, a neke datoteke posjeduju više od jedne neispravnosti. Značaj parametra 6.2. analizira se usporedbom broja povezanih neispravnosti, ukupnog broja datoteka te utvrđenih DN parova. Parametar 6.3. je posljedica toga što jedna datoteka može biti povezana s više predaja, koje su pak povezane s istom neispravnošću. Njezin značaj se utvrđuje prebrojavanjem dupliciranih DN parova.

Tablica 3.9 pokazuje da je broj DN parova veći od broja datoteka i od broja povezanih neispravnosti, što potvrđuje da je kardinalnost njihove veze više-na-više. Također, prikazuje i broj

Tablica 3.9: *Parametri 6.2. i 6.3.* - veza datoteka i neispravnost

Broj povezanih	**Višestruke veze po projektima**				
	JDT	PDE	Platform	BIRT	Mylyn
Neispravnosti:	13468	3995	24663	1532	441
Datoteka:	12080	6629	15084	4273	975
DN parova:	25476	16595	23784	6663	1258
Duplih DN parova:	768	1233	903	4	0

duplih DN parova, za koje je potrebno ustanoviti uzrok pojavljivanja. Dodatnom ručnom analizom utvrđeno je da neki DN parovi mogu biti duplicirani više od jednom. Uzrok dupliciranih parova je to što za isti DN par postoji:

- Predaja s testom i predaja s ispravljenim kodom;
- 2 predaje koje ispravljaju kod;
- 1 ili 2 predaje koje predstavljaju privremeni ispravak;
- Ispravak i *povratak* (eng. revert) na staru inačicu.

Dodatnom analizom ustanovljeno je da se u svim slučajevima samo posljednja predaja može smatrati relevantnom za ispravljanje koda. Problem dupliciranih parova datoteka i neispravnosti nije komentiran u srodnoj literaturi. U nastavku ove disertacije, u poglavlju 5.2.2, provedena je ručna usporedba podataka dobivenih SZZ pristupom. Ta analiza utvrdila je da SZZ pristup, iako to nigdje izričito ne navodi, uzima u obzir duplicirane DN parove te da ih broji samo jednom.

3.3 Diskusija rezultata

U ovom poglavlju predstavljen je sustavni pregled literature te istraživačka studija slučaja čiji je cilj bio otkriti sve parametre koji mogu utjecati na ishod prikupljanja podataka za SDP. Zaključci iz ovog poglavlja omogućit će sustavnu definiciju procedure prikupljanja podataka za SDP te izgradnju alata za njezino automatizirano provođenje. Novi alat i tehnika za povezivanje repozitorija iz sustava za praćenje zahtjeva i upravljanje inačicama predstavljeni su u idućem poglavlju.

Iz sustavnog pregleda literature može se zaključiti da postoji mnogo radova na temu SDP-a. Međutim, ne postoji sustavno opisan postupak prikupljanja podataka. Iako postoje neke norme, poput norme za unos neispravnosti u sustav za praćenje zahtjeva IEEE 1044-2009, one se ne

spominju u srodnoj literaturi. Umjesto toga, postoje postupci za prikupljanje podataka koji nisu verificirani niti međusobno uspoređeni. U takvom okruženju, novija istraživanja nerijetko se priklone najčešće korištenim postupcima bez kritičkog osvrta na njih. Sustavnim pregledom literature ustanovljeno je da se postojeći postupci međusobno razlikuju u odabiru parametara prikupljanja podataka. Stoga, ugrožena je valjanost konstrukcije onih istraživanja koja su prikupljala podatke na različite načine.

Istraživačka studija ukazala je na parametre prikupljanja podataka koji nisu bili spomenuti u srodnoj literaturi. To otkriva da je ugrožena i valjanost zaključaka onih istraživanja koja su prikupljala podatke nedovoljno precizno definiranim postupkom koji nije moguće u potpunosti ponoviti. Utjecaj pojedinih parametara je i kvantificiran, tako što se analiziralo koliko bi podaci mogli biti drugačiji ukoliko bi se njihove vrijednosti ostavilo otvorenima interpretaciji.

Valjanost istraživanja

Valjanost sustavnog pregleda literature je ograničena s obzirom da su korištene samo dvije baze znanstveno-istraživačkih radova, IEEE Xplore i ACM Digital Library. Međutim, to su ujedno i dvije najreferentnije baze, koje sadrže najvažnije radove u ovome području. Valjanost konstrukcije, odnosno ispravnost korištene metode i valjanost zaključaka, odnosno njihova ponovljivost, nisu narušeni. Dosljedno su se pratile preporuke za sustavni pregled literature u području programskog inženjerstva i konkretna primjena tih preporuka u sustavnom pregledu literature, koji je proveden za razdoblje do 2010. godine. Vanjska valjanost, odnosno mogućnost poopćenja rezultata donekle je ograničena činjenicom da je većina pregledanih radova provodila istraživanje u domeni projekata otvorenog koda, a mnogo manje u industrijskom okruženju.

Isto vrijedi i za istraživačku studiju slučaja. Što se tiče valjanosti konstrukcije, korišten je mali broj izvora podataka, ali su podaci vrlo reprezentativni. Riječ je o dugovječnim i složenim projektima koji su nerijetko bili predmet istraživanja i u srodnim studijama. Vanjska valjanost ograničena je na domenu tih projekata, odnosno na projekte otvorenog koda. Budući da su industrijski projekti uvijek zatvorenog karaktera, nije bilo mogućnosti provedbe istraživanja i u tom okruženju. Valjanost zaključaka unaprijeđena je primjenom tehnike triangulacije promatrača te pomnom pripremom seminara, zadataka i uputa za provedbu istraživanja.

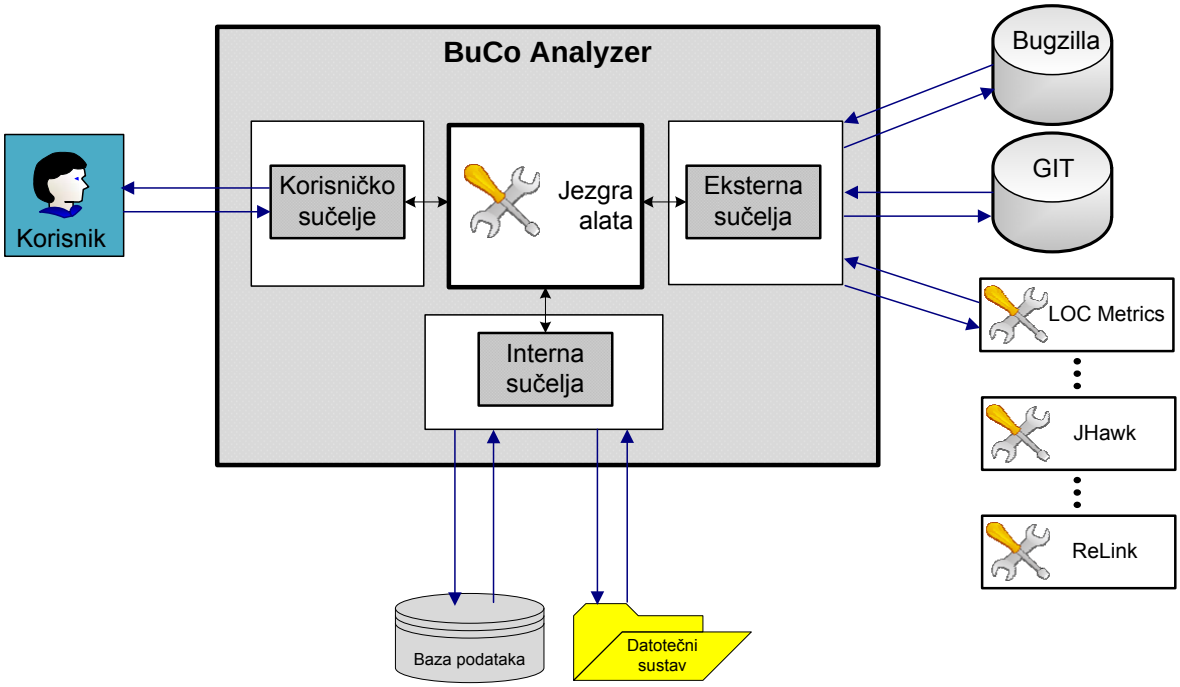
Poglavlje 4

Alat i nova tehnika za prikupljanje podataka

Empirijska istraživanja, na kojima se u velikoj mjeri temelji otkrivanje novih spoznaja u programskom inženjerstvu, zahtijevaju veliku količinu podataka prikupljenih na konzistentan način. Najbolje rješenje za taj zahtjev pronađeno je u izradi alata za automatizirano prikupljanje podataka koji se temelji na sustavno definiranim smjernicama. Temeljem vrijednih spoznaja o parametrima prikupljanja podataka za SDP, predstavljenih u prethodnom poglavlju, izrađen je upravo takav alat. Alat koji povezuje neispravnosti (eng. bug) iz sustava za praćenje zahtjeva Bugzilla i programski kod (eng. code) iz sustava za upravljanje inačicama GIT nazvan je Bug-Code Analyzer (BuCo). Akronim njegova imena (BuCo) skriva i činjenicu da je sposoban analizirati i pohraniti velike količine podataka u svoju internu bazu podataka.

4.1 Alat za automatizirano prikupljanje podataka

Predviđanje lokacija neispravnosti unutar izvornog koda temelji se na podacima o prijašnjim iskustvima. Glavna prepreka korištenju predviđanja je zahtjevan proces prikupljanja takvih podataka. Potrebno je prikupiti veliku količinu podataka iz formalno nepovezanih razvojnih repozitorija, što zahtijeva mnogo truda i vremena te je kao takvo preskupo u većini komercijalnih projekata [45]. Povezivanje razvojnih repozitorija moguće je provesti uporabom jedne ili kombinacije više različitih tehnika predstavljenih u poglavlju 2.1.1. Automatizacija navedenog procesa u formi nekog alata slijedi kao evidentna potreba. Kako nije pronađen niti jedan alat koji zadovoljava sve te uvjete, razvijen je vlastiti. Alat BuCo razvijen je iterativno. Unaprijeđen



Slika 4.1: Arhitektura alata BuCo

je iterativno svim stečenim spoznajama koje su predstavljene u ovoj disertaciji. Alat je razvijen koristeći pouzdane tehnologije - server Apache HTTP, relacijsku bazu podataka MySQL, repozitorij za upravljanje inačicama sustava GIT i programski jezik Python.

4.1.1 Arhitektura alata BuCo

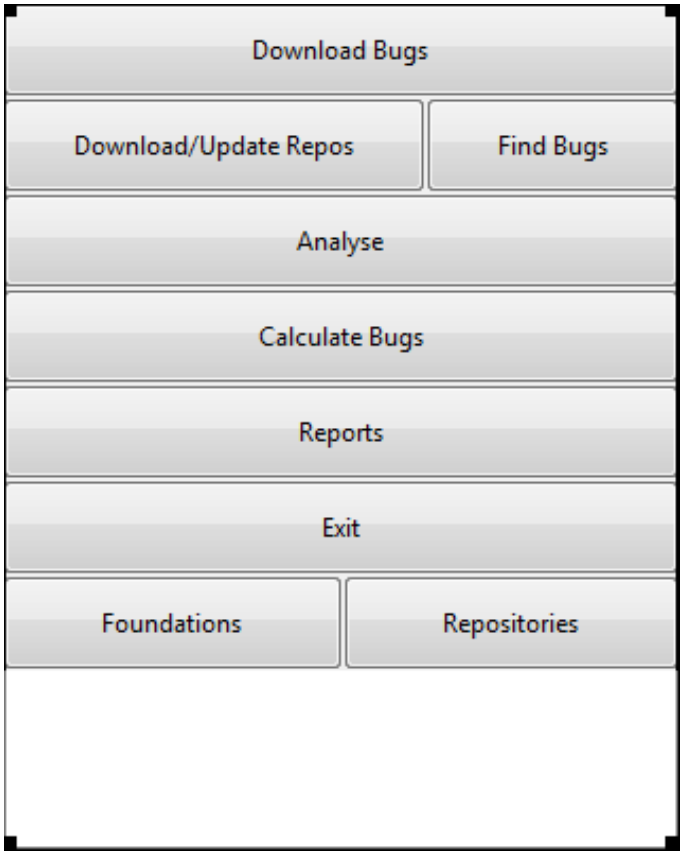
Alat BuCo implementira sučelje prema razvojnim repozitorijima, alatima za metrike, alatu ReLink za povezivanje neispravnosti i predaja, te grafičko korisničko sučelje za jednostavniju uporabu. Arhitektura alata prikazana je slikom 4.1.

Razvojni repozitoriji prema kojima trenutna inačica alata ima sučelje jesu repozitorij za upravljanje inačicama sustava GIT i repozitorij za praćenje zahtjeva sustava Bugzilla. Navedeni sustavi odabrani su jer se koriste u vodećim razvojnim zajednicama projekata otvorenog koda poput Apache, Eclipse i Mozilla. Alati za metrike LOC Metrics i JHawk odabrani su istraživačkom studijom koja je predstavljana u poglavlju 3.2. ReLink je alat koji povezuje neispravnosti i predaje, a pritom koristi tehnike jednostavne pretrage, podudaranja autorstva i vremenske korelacije, obrade prirodnog jezika, te napredne tehnike predviđanja veza koje nedostaju [92]. Sučelje s datotečnim sustavom služi za pohranu i analizu cjelokupnih repozitorija iz sustava GIT te rad s izvješćima alata za metrike. Sučelje s bazom podataka služi za pohranu

svih relevantnih podataka za: razvojne zajednice, karakteristike prikupljenih neispravnosti, inačice projekata, otkrivene veze između neispravnosti i predaja, karakteristike povezanih predaja, te metrike i broj neispravnosti analiziranih datoteka. Korisničko sučelje čini alat uporabljivim čak i za korisnika koji ne poznaje sve pojedinosti pozadinskih radnji koje su nužne za uspješno prikupljanje podataka. Početni prozor korisničkog zaslona alata BuCo, koji otkriva i osnovne funkcionalnosti alata, prikazan je slikom 4.2. Detaljniji opis funkcija alata slijedi u narednom poglavlju.

4.1.2 Funkcionije alata BuCo

Prilikom prvog pokretanja alata, pojavljuje se konfiguracijski prozor za unos imena baze podataka te korisničkog imena i lozinke korisnika baze. Unesene vrijednosti zapisuju se u konfiguracijsku datoteku config.ini, gdje se mogu promijeniti ukoliko zatreba. Alat potom omogućuje unos imena razvojne zajednice i poveznice na njezin sustav za praćenje neispravnosti, gumbom *Foundations*. Omogućen je i unos poveznica na repozitorije željenih projekata iz sustava za upravljanje inačicama GIT, gumbom *Repositories*. Nakon ovih inicijalnih koraka, korištenje



Slika 4.2: Početni prozor alata BuCo

alata slijedi *odozgo-prema-dolje* (eng. top-down) logiku redoslijeda uporabe gumba prikazanih slikom 4.2.

Prikupljanje neispravnosti

Prikupljanje neispravnosti pokreće se gumbom *Download Bugs*. Omogućeno je prikupljanje neispravnosti i njihovih karakteristika iz repozitorija sustava za praćenje zahtjeva Bugzilla za razvojne zajednice Eclipse, Mozilla i Apache. Iako navedene razvojne zajednice imaju neke razlike u pristupu neispravnostima, alat ih rješava u pozadini. Korisnik odabire za koji projekt i za koje njegove inačice želi prikupiti neispravnosti. Odabirom sljedećih parametara dobiju se neispravnosti koje su uzrokovale gubitak funkcionalnosti prije nego su razriješene i zatvorene nakon što su uspješno ispravljene (parametar 3.1 i 3.2 iz poglavlja 3.2.1):

- **Status:** razrješen, verificiran ili zatvoren;
- **Rezolucija:** ispravljena;
- **Težina:** blokirajuća, kritična, velika, normalna ili minorna.

Prikupljanje izvornog koda

Prikupljanje izvornog koda pokreće se gumbom *Download/Update Repos*. Za projekte za koje se preuzmu neispravnosti te se unesu poveznice na sve podrepozitorije, pokreće se preuzimanje cjelovitih GIT repozitorija (parametar 4.1 iz poglavlja 3.2.1). Pri tome se oznake inačice projekata iz sustava za praćenje zahtjeva povezuju s odgovarajućim oznakama unutar sustava za upravljanje inačicama (parametar 2.2 iz poglavlja 3.2.1). Također, u bazu se unosi i datum predaje svake od traženih inačica. Ukoliko je repozitorij prethodno već bio pohranjen, onda se samo osvježava s najnovijim predajama. Ukoliko repozitorij odabranog projekta ne postoji u sustavu za upravljanje inačicama, korisnik dobiva odgovarajuću poruku (parametar 2.1 iz poglavlja 3.2.1). Preuzeti repozitoriji sadrže kompletnu povijest izmjene izvornog koda i svih njezinih detalja. Svaki podrepozitorij se pohranjuje u zaseban direktorij, a ne postoji ograničenje njihove veličine. Veličina repozitorija može utjecati jedino na brzinu provedbe povezivanja neispravnosti i predaja.

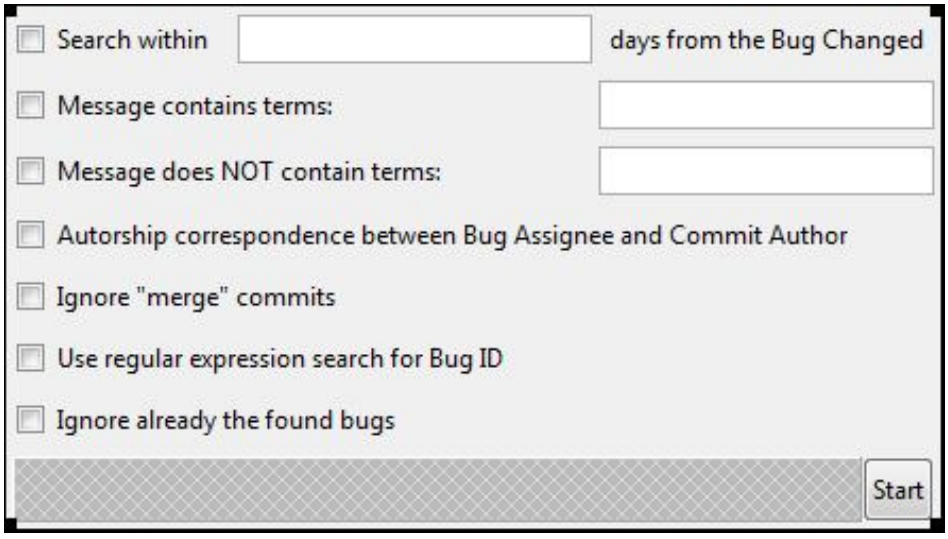
Povezivanje neispravnosti i predaja

Povezivanje neispravnosti i predaja pokreće se gumbom *Find Bugs*. Budući da postoji niz tehnika kojima se to može postići, BuCo alat otvara novi prozor u kojem je moguće odabrati jednu

ili kombinaciju više tehnika. Odabir tehnika prikazan je slikom 4.3, a one su sljedeće:

- Jednostavna pretraga (ukoliko se ne odabere niti jedna druga tehnika);
- Vremenska korelacija u kojoj se može definirati ciljani raspon dana od datuma kada je neispravnost zatvorena (eng. *search within [prazno mjesto za upis broja] days from the Bug Changed*);
- Pretraga ključnih izraza u kojoj se može navesti jedna ili više riječi koje opis predaje mora sadržavati (eng. *message contains terms*) ili ne smije sadržavati (eng. *message does NOT contain terms*) da bi bio povezan;
- Podudaranje autorstva (eng. *authorship correspondence between Bug Assignee and Commit Author*);
- Pretraga uz dodatnu opciju koja preskače predaje *spajanja* (eng. *ignore merge commits*);
- Pretraga regularnim izrazom koja definira okružje ID-a neispravnosti (eng. *use regular expression search for Bug ID*);
- Pretraga uz dodatnu opciju koja preskače već povezane neispravnosti (eng. *ignore the already found bugs*).

Dodatna opcija koja preskače predaje spajanja nastala je nakon što se ručnom analizom utvrdilo da one često uzrokuju duplicirane poveznice. Pretraga regularnim izrazom nova je tehnika, koja je nastala kao produkt istraživačke studije iz prethodnog poglavlja i analize slučaja, koja je predstavljena u idućem poglavlju. Nova tehnika zadovoljava parametar 5.1 iz poglavlja 3.2.1. Opcija preskakanja već povezanih neispravnosti omogućuje jednostavno povezivanje ukoliko



Slika 4.3: Prozor alata BuCo za odabir tehnike povezivanja

se neke tehnike žele uzastopno pokretati. Ono što vrijedi za sve tehnike je da će pohraniti sve pronađene veze u bazu podataka koja predviđa vezu više-na-više (parametar 5.2 iz poglavlja 3.2.1). Pored toga, u bazu će se pohraniti i datum predaje, putanje izmijenjenih datoteka, te količina izmijenjenih i izbrisanih linija koda za svaku datoteku.

Izračun broja neispravnosti

Izračun broja neispravnosti za svaku datoteku iz određene inačice projekta otvorenog koda pokreće se gumbom *Calculate Bugs*. Ovo je automatizirani korak koji pretvara veze između neispravnosti i predaje u veze neispravnosti i datoteka. To je ostvareno preko veza između predaja i datoteka koje su pohranjene u bazu podataka istovremeno s vezama neispravnosti i predaja. Veza neispravnosti i datoteka može biti kardinalnosti više-na-više (parametar 6.2 iz poglavlja 3.2.1). Prilikom izračuna broja neispravnosti, za slučaj da postoji višestruka veza neke neispravnosti i predaje, uzima se u obzir samo posljednja (parametar 6.3 iz poglavlja 3.2.1).

Analiza metrika

Analiza metrika pokreće se gumbom *Analyse*. Ovaj korak ne nudi korisniku dodatne opcije, već se automatizirano provodi:

- Slijedni *dohvat* (eng. check out) odabranih inačica projekata iz preuzetih repozitorija sustava GIT koristeći prethodno dohvaćene oznake;
- Pokretanje alata LOC Metrics za izračun metrika (navedenih u tablici P.1 u prilogu P1) svih datoteka s ekstenzijom *.java* te njihova pohrana u bazu podataka;
- Pokretanje alata JHawk za izračun metrika (navedenih u tablici P.2 u prilogu P1) svih prethodno analiziranih datoteka s ekstenzijom *.java*;
- Povezivanje metrika oba alata koristeći isti format putanje datoteka kao i u koraku povezivanja neispravnosti i predaja (parametar 6.1 iz poglavlja 3.2.1);
- Analiza koja provjerava postoje li datoteke u idućoj inačici istog projekta te jesu li pri tome mijenjane;
- Unos svih izračunatih metrika i rezultata analiza u bazu podataka, ali odvojeno za svaku inačicu projekta.

Alati LOC Metrics i JHawk zadovoljavaju parametar 1.1 iz poglavlja 3.2.1. Analiza koja provjerava postoje li datoteke u idućoj inačici te jesu li mijenjane može poslužiti u dodatnim pretragama i istraživanju ispravljenih neispravnosti kod kojih nedostaju poveznice s predajama.

Detaljnija diskusija o metrikama, njihovoj važnosti i opisu onih koje računa alat Buco nalazi se u prilogu P1.

Izrada izvješća

Izrada gotovih skupova podataka za SDP pokreće se gumbom *Reports*. To je posljednja faza u prikupljanju podataka koja se pokreće tek nakon što su sve prethodne obavljene. Korisniku se omogućuje izrada nekoliko vrsta izvješća:

- Gotovi skupovi podataka za SDP;
- Općenita izvješća o količini analiziranih podataka;
- Specifična izvješća za pojedinu metriku;
- Deskriptivna statistička izvješća za metrike.

Gotovi skupovi podataka za SDP su osnovni produkt i svrha ovog alata, a odgovaraju Formi 5 iz poglavlja 3.2. Generalna izvješća prikazuju količinu analiziranih projekata, neispravnosti i datoteka koje sadrže njihove inačice, te broj uspješno povezanih neispravnosti i datoteka. U specifičnim izvješćima može se tražiti konkretna vrijednost za pojedinu metriku. Za odabrane metrike, deskriptivna statistička izvješća računaju minimalnu, maksimalnu i srednju vrijednost, medijan i interkvartilni raspon, provjeravaju pripadaju li njezine vrijednosti normalnoj distribuciji uporabom odgovarajućeg statističkog testa te računaju korelacijsku matricu.

4.2 Tehnika povezivanja temeljena na regularnim izrazima

Tehnike povezivanja repozitorija sustava za praćenje zahtjeva i sustava za upravljanje inačicama razlikuju se s obzirom na to koje podatke koriste za otkrivanje veza. No, gotovo sve procedure slažu se u uporabi *ID-a neispravnosti* (eng. Bug ID) kao najvjerodostojnije poveznice. Na tome se temelji i tehnika jednostavne pretrage. Istraživačka studija, predstavljena u poglavlju 3.2 pokazala je da uporaba te tehnike može dovesti do neispravnih podataka. Uporaba naprednih algoritama za uspostavu i predviđanje veza, poput onih u alatu ReLink, nudi se kao jedno od rješenja tog problema [92]. Međutim, njihova učinkovitost je osporena analizom koja je provedena nad etalonima skupova podataka [93]. U ovom poglavlju predstavljena je analiza slučaja čija je svrha unapređenje spomenutih tehnika povezivanja. Sama analiza slučaja podijeljena je u tri dijela. Cilj prvog dijela ove analize slučaja je identifikacija problema tih tehnika. Nakon analize rezultata prvog dijela predložena je nova tehnika povezivanja temeljena na regularnim izrazima. Cilj drugog i trećeg dijela ove analize slučaja je provjera učinkovitosti nove tehnike. Nova tehnika uspoređena je s jednostavnom pretragom u drugom dijelu, odnosno s alatom Relink u trećem dijelu. Radi jednostavnije i konzistentnije provedbe ove analize slučaja, sve tehnike ugrađene su u alat BuCo. Nova tehnika svojstvena je samo alatu BuCo i temeljena na regularnim izrazima pa se u nastavku disertacije naziva BuCo Regex.

4.2.1 Metodologija analize slučaja

Analiza slučaja prati preporuke za provedbu analize slučaja [28]. Radi višestrukosti izvora podataka korišteni su podaci iz dva izvora: podaci iz projekta Apache HTTPD koji su priloženi uz alat ReLink te etalon skupova podataka iz projekta Apache OpenNLP. Autori alata ReLink postigli su izvrsne rezultate upravo s podacima iz projekta Apache HTTPD. Zajedno s alatom mogu se preuzeti njegovi ulazni podaci o neispravnostima i izmjenama te izlazni podaci koji otkrivaju njihove veze. Upravo u toj domeni uspoređen je alat ReLink s jednostavnom pretragom, kako bi se stekao uvid u njihove prednosti i nedostatke. Ručna analiza dobivenih rezultata poslužila je kao temelj za izgradnju nove tehnike BuCo Regex, koja je trebala nadvladati nedostatke iz obje prethodno uporabljene tehnike. Kritika na učinkovitost alata ReLink temeljena je na etalonu skupova podataka koji su javno dostupni. Jedan takav skup pripremljen je za projekt Apache OpenNLP pa je i u toj domeni provedena nepristrana usporedba tehnike BuCo Regex i alata ReLink. Dakle, dizajn ove analize slučaja podrazumijeva sljedeće dijelove:

(1) **Analiza 1:** Usporedba alata ReLink i tehnike jednostavne pretrage nad podacima iz projekta Apache HTTPD;
(2) Vrednovanje dobivenih rezultata, ručna istraga netočnih veza i izrada kriterija za pretragu regularnim izrazima;
(3) **Analiza 2:** Usporedba alata ReLink i tehnike BuCo Regex nad podacima iz projekta Apache HTTPD;
(4) **Analiza 3:** Usporedba alata ReLink i tehnike BuCo Regex nad etalonom skupa podataka iz projekta Apache OpenNLP.

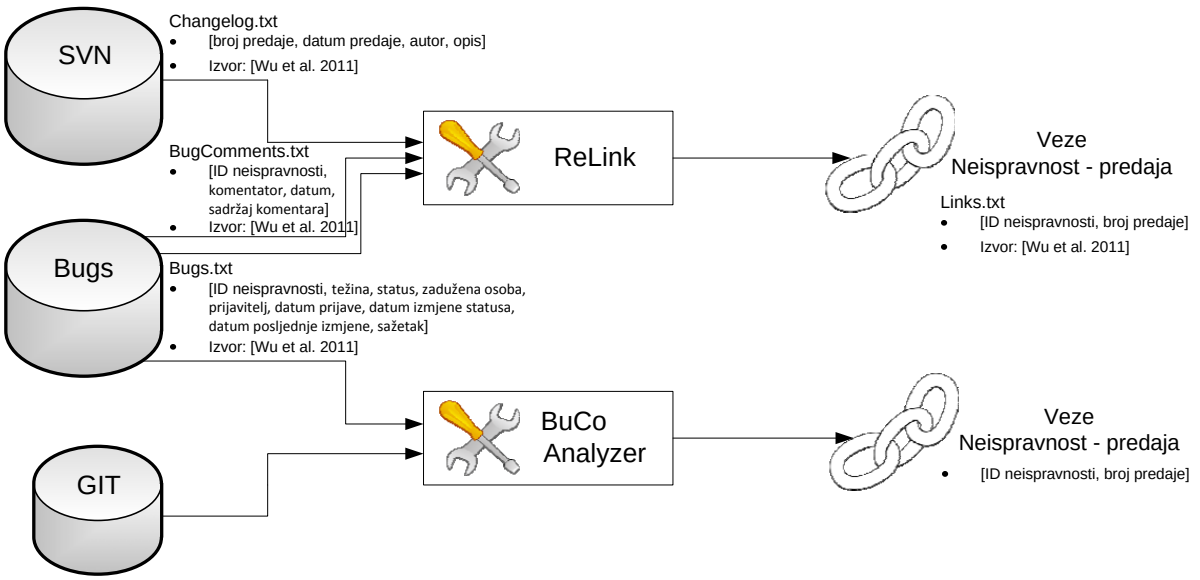
Materijal analize slučaja

Za provedbu ove analize slučaja koristi se alat BuCo, koji implementira sučelje prema alatu ReLink te ima ugrađenu jednostavnu pretragu i tehniku BuCo Regex. Alat ReLink detaljnije je obrađen u poglavlju 2.1.1. Važno je napomenuti da alat ReLink i alat BuCo ne koriste isti sustav za upravljanje inačicama. Alat ReLink namijenjen je za rad sa sustavom za upravljanje inačicama SVN, dok alat BuCo radi sa sustavom GIT, koji je inače šire zastupljen. Radi njihove kompatibilnosti, alat BuCo napravljen je tako da može pripremiti podatke iz formata sustava GIT u format sustava SVN, kakav zahtijeva alat ReLink.

Dizajn analize slučaja

Provedba analize 1 zasniva se na podacima iz projekta Apache HTTPD koji su pohranjeni u datotekama *Bugs.txt*, *BugComments.txt*, *Changelog.txt* i *Links.txt* te u repozitoriju tog projekta iz sustava GIT. Sve navedene datoteke preuzete su zajedno s alatom ReLink*. Datoteka *Bugs.txt* sadrži popis neispravnosti te njihove osnovne karakteristike: ID, težina, status, zadužena osoba, prijavitelj, datum prijave, datum izmjene statusa, datum posljednje izmjene i sažetak. Datoteka *BugComments.txt* sadrži listu svih komentara za neispravnosti u formatu: ID neispravnosti, autor komentara, datum i sadržaj komentara. Datoteka *Changelog.txt* sadrži informacije o predajama izmjena programskog koda u formatu sustava SVN: broj predaje, datum predaje, autor i opis. Te tri datoteke su ulazni podaci za alat ReLink na temelju kojih on povezuje neispravnosti i predaje. Datoteka *Links.txt* sadrži popis veza koje je otkrio alat ReLink u formatu ID neispravnosti i broj predaje. Radi što većeg podudaranja, za jednostavnu pretragu korišten je popis neispravnosti iz datoteke *Bugs.txt*. Budući da alat BuCo ne može raditi s repozitorijem

*https://code.google.com/p/bugcenter/wiki/ReLink



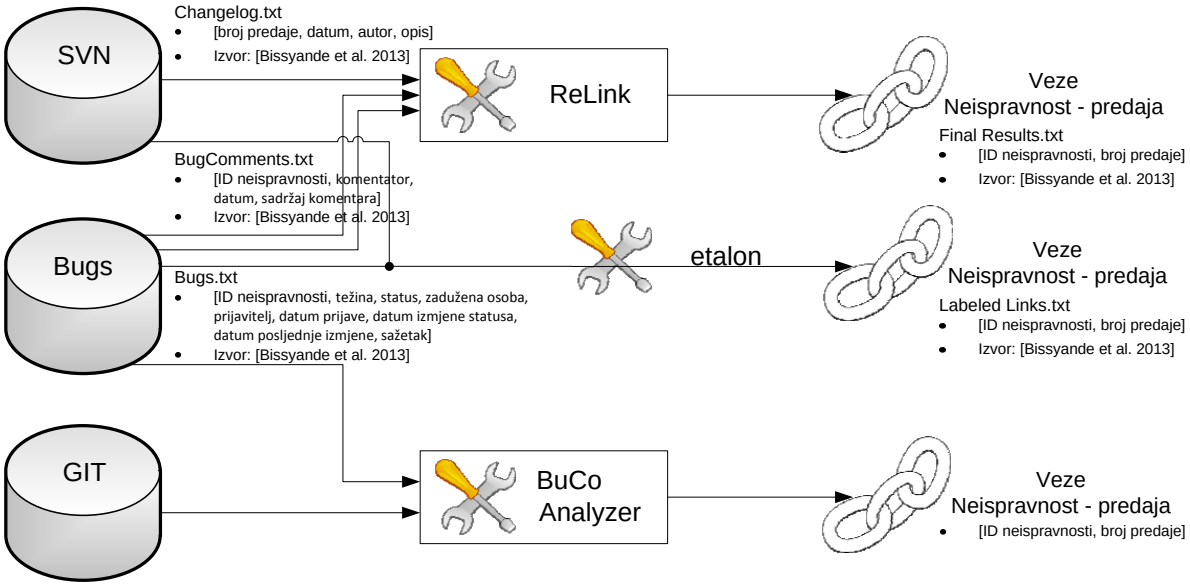
Slika 4.4: Izvor i struktura ulaznih podataka za analizu 1 i analizu 2

u formatu sustava SVN, nije bilo moguće uporabiti datoteku *Changelog.txt*. Međutim, projekt Apache HTTPD je migrirao na sustav GIT pa je njegov repozitorij preuzet sa servisa Github[†].

Podudarnost i ispravnost veza koje su pronašli alat ReLink i tehnika jednostavne pretrage ispituje se nakon provedene analize 1. Cilj je ustanoviti u kojoj mjeri jednostavna pretraga i alat ReLink daju iste rezultate, jesu li oni ispravni, te u kojoj mjeri jednostavna pretraga griješi. Ispravnost uspostavljenih veza ispituje se ručnom pretragom. Naročita pažnja pridaje se otkrivanju konzistentnih pogrešaka i pravilnih uzoraka koji bi se novom tehnikom BuCo Regex mogli ispraviti. Nakon što se definira kriterij pretrage za tehniku BuCo Regex, provodi se analiza 2. Analiza 2 je istovjetna analizi 1, uz razliku da se umjesto jednostavne pretrage koristi tehnika BuCo Regex. Sažeti prikaz toka podataka analize 1 i analize 2 prikazan je slikom 4.4. Slika 4.4 grafički ukazuje i na razliku u veličini repozitorija iz sustava za upravljanje inačicama SVN i GIT koji pripadaju projektu Apache HTTPD.

Analiza 3 uspoređuje alat ReLink i tehniku BuCo Regex u kontroliranim uvjetima etalona skupova podataka za projekt Apache OpenNLP. Podaci su preuzeti iz repozitorija etalona skupova podataka[‡]. U tom repozitoriju nalaze se svi ulazni podaci u datotekama analognima ulaznim datotekama iz prethodne analize: *Bugs.txt*, *BugComments.txt* i *Changelog.txt*. Svrha tih podataka je provedba analiza slučaja poput ove, u kojima treba ispitati učinkovitost tehnika za povezivanje repozitorija sustava za praćenje zahtjeva i sustava za upravljanje inačicama. Etalon stoga sadrži izlaznu datoteku *Labeled Links.txt* koja otkriva sve postojeće poveznice između

[†]https://github.com/apache/httpd.git
[‡]http://momentum.labri.fr/bugLinking/



Slika 4.5: Izvor i struktura ulaznih podataka za Analizu 3

neispravnosti pohranjenih u datoteci Bugs.txt i predaja pohranjenih u datoteci Changelog.txt. Također, sadrži i izlazne datoteke od alata ReLink pod imenom *Final Results.txt*. Za pokretanje tehnike BuCo Regex, preuzet je repozitorij projekta OpenNLP sa servisa Github[§]. Za razliku od prethodne dvije analize, repozitorij iz sustava GIT bio je iste veličine i sadržaja kao i repozitorij iz sustava SVN. U analizi 3 uspoređuje se podudarnost i ispravnost veza koje pronađu alat ReLink i tehnika BuCo Regex s vezama koje su prisutne u etalonu skupa podataka. Sažeti prikaz toka podataka analize 3 prikazan je slikom 4.5.

Za analizu rezultata i usporedbu različitih tehnika povezivanja definirane su sljedeće mjere ulaznih i izlaznih podataka:

- Ulaz: broj predaja iz sustava za upravljanje inačicama (BP_u);
- Ulaz: broj neispravnosti iz sustava za praćenje zahtjeva (BN_u);
- Izlaz: broj uspostavljenih Veza (BV_i);
- Izlaz: broj povezanih Predaja (BP_i);
- Izlaz: broj povezanih Datoteka (BD_i);
- Izlaz: broj povezanih Neispravnosti (BN_i);
- Izlaz: stopa povezivanja (SP).

Mjere ulaznih podataka BP_u i BN_u služe za provjeru jesu li tehnike primijenjene u jednakim uvjetima. Mjere izlaznih podataka služe za usporedbu uspješnosti tehnika povezivanja. Tehnika, pritom, nije nužno uspješnija ukoliko ima veću vrijednost izlaznih mjera. Razlike u dobi-

[§]https://github.com/apache/opennlp

venim vrijednostima izlaznih mjera mogu samo ukazati na odstupanje. Točan uzrok odstupanja kasnije se treba istražiti ručnom pretragom. Točnost pronađenih veza utvrđuje se usporedbom s etalonom skupa podataka ili ručnom pretragom ukoliko etalon ne postoji.

4.2.2 Rezultati analize slučaja

Rezultati analize 1 prikazani su u tablici 4.1. Razlika u ulaznim podacima koja je naglašena slikom 4.4 evidentna je i u broju predaja BP_u. Uzrok te razlike je nepotpuna migracija projekta iz sustava SVN u sustav GIT. Posljedica te razlike je da alat ReLink uspostavi određeni broj veza prema predajama koje nisu postojale za jednostavnu pretragu. S druge strane, ulazni podaci za neispravnosti su identični jer za obje tehnike potječu iz datoteke *Bugs.txt*. Izlazni podaci ukazuju na veliku razliku u broju uspostavljenih veza BV_i, odnosno povezanih predaja BP_i i datoteka BD_i. Tehnika jednostavne pretrage ne uspijeva povezati sve neispravnosti što uzrokuje manju stopu povezivanja *SP*. Alat ReLink, s druge strane, uspijeva povezati sve neispravnosti i ostvariti stopu povezivanja od 100%.

Ručna istraga rezultata iz analize 1 analizirala je isključivo izlazne podatke, odnosno nije tragala za eventualnim vezama koje obje tehnike nisu uspjele pronaći. Svaka veza koju je ostvarila jednostavna pretraga potražena je u skupu veza koje je ostvario ReLink i validirana je njezina ispravnost. Tablica 4.2 prikazuje rezultate provedene istrage. Broj veza koje su zajedničke za obje tehnike je 443, odnosno 74.1% veza koje je pronašla jednostavna pretraga. Sve navedene veze pokazale su se ispravnima. Među podudarajućim vezama otkriveno je da njih 196 predstavlja jedinu poveznicu za neke neispravnosti. Sve neispravnosti koje su na taj način povezane imaju isti rezultat za obje tehnike. Otkriveno je i da 247 veza predstavlja višestruke poveznice za neke neispravnosti. Neke neispravnosti su ispravno povezane s više različitih predaja te to čini 247 podudarajućih veza. Međutim, ReLink pronalazi i neke duplicirane veze između jedne neispravnosti i jedne predaje. Alat ReLink sadrži duplicirane veze i za predaje spajanja (eng. merge) koje se nalaze u repozitoriju sustava SVN, ali ne i u repozitoriju sus-

Tablica 4.1: Rezultati povezivanja za (1) ReLink i (2) jednostavna pretraga

Analiza	**Ulazni podaci**			**Tehnika Povezivanja**	**Izlazni podaci**				
	Izvor	BP_u	BN_u		BV_i	BP_i	BD_i	BN_i	*SP*
1	SVN + Bugs.txt	43,867	673	**(1)**	1014	957	1061	673	100%
	GIT + Bugs.txt	26,287	673	**(2)**	598	556	993	598	89%

Tablica 4.2: Ručna istraga veza tehnike jednostavna pretraga

Usporedba izlaznih podataka	Broj	Postotak
Podudarajuće veze	443	74.1%
Neispravnosti sa jednostrukom vezom	196	32.8%
Neispravnosti sa višestrukom vezom	247	41.3%
Različite veze	147	24.6%
Neispravno uspostavljene veze	120	20.1%
Veze dvojbene ispravnosti	27	4.5%
Veze na GIT predaje kojih nema u SVN-u	8	1.3%

tava GIT. Predaje spajanja sažimaju sve predaje koje je neki programer učinio kada je radio na izmjenama koda u odvojenoj grani (eng. branch) unutar sustava za upravljanje inačicama. Takve duplicirane veze uzrokuju veliku razliku u broju ostvarenih veza BV_i i povezanih predaja BP_i. Istovremeno uzrokuju i mnogo manju razliku u broju povezanih datoteka BD_i jer samo ponavljaju koje su datoteke mijenjane u predajama koje predaja spajanja sažima. Razliku u broju povezanih datoteka BD_i umanjuju i veze koje je ostvarila samo jednostavna pretraga. ReLink povezuje neke neispravnosti koje jednostavna pretraga ne povezuje te to uzrokuje razliku u broju povezanih neispravnosti BN_i između ove dvije tehnike. Broj veza koje je pronašla jednostavna pretraga, a ReLink nije iznosi 147, što čini 24.6% od ukupnog broja veza jednostavne pretrage. Od toga, čak 120 veza je neispravnih te one predstavljaju temelj izgradnje tehnike BuCo Regex. Za 27 veza nije bilo moguće ustvrditi ispravnost bez pomoći samih programera koji su ih učinili. Jednostavna pretraga je pronašla i 8 ispravnih veza koje ReLink nije mogao pronaći jer se povezane predaje nisu nalazile u repozitoriju sustava SVN.

Ručna istraga neispravnih veza ustanovila je da se ID neispravnosti gotovo uvijek nalazi uz neki drugi alfanumerički znak. Zbog toga, ID neispravnosti koji tražimo neispravno se pronalazio u ID oznaci neispravnosti koja ima više znamenaka, ID oznaci nekog drugog entiteta (npr. predaje) ili općenito u drugim brojčanim zapisima. S druge strane, u određenom broju ispravnih veza ustanovilo se da se broj ID oznake može nalaziti uz neku riječ koja mu prethodi (npr. *bug*, *fix* ili *pr*) ili kao početak niza znakova. Stečena saznanja pretočena su u unaprijeđeni kriterij pretrage, koji se temelji na sljedećem regularnom izrazu:

$$'(.*[\hat{}0-9]|\hat{})' + bug\_id + '(\backslash W|\backslash\backslash r|\$)' \tag{4.1}$$

Regularni izraz 4.1 definira sljedeće:

- Znak koji prethodi ID oznaci neispravnosti može biti početak niza znakova ili bilo koji znak koji nije numerički;
- Broj ID oznake neispravnosti mora se u potpunosti podudarati;
- Znak koji slijedi iza ID oznake neispravnosti može biti znak kraj niza znakova ili bilo koji znak koji nije alfanumerički.

Rezultati analize 2 i analize 3 prikazani su u tablici 4.3. Kao i u prethodnom slučaju, analiza 2 ima nejednake ulazne podatke za projekt Apache HTTPD. Analiza 3, s druge strane, ima jednake ulazne podatke, iako također koristi repozitorije različitih sustava za upravljanje inačicama. Za projekt Apache OpenNLP, repozitoriji sustava SVN i GIT imaju jednaki broj predaja BP_u te čine dobar temelj za usporedbu. Pored toga, za analizu 3 koristi se i etalon skupova podataka koji za ponuđenih 100 neispravnosti nudi sve veze koje bi trebalo otkriti.

Analiza 2 otkriva da se uporabom tehnike BuCo Regex može povezati veći broj neispravnosti BN_i, te ostvariti veća stopa povezivanja *SP* i više veza BV_i nego li uporabom jednostavne pretrage. BuCo Regex ostvaruje stopu povezivanja od čak 93%, ali to je i dalje manje od alata ReLink koji ostvaruje 100%. Ručnom istragom veza koje je uspostavio ReLink, a nije BuCo Regex, ustanovljeno je da sve povezane predaje sadrže ID neispravnosti unutar svoga opisa te da se povezane predaje ne nalaze u repozitoriju sustava GIT. Drugim riječima, tehnika BuCo Regex pronašla bi te predaje ukoliko bi se nalazile u repozitoriju sustava GIT i ostvarila jednak uspjeh kao i ReLink. Važno je napomenuti da ručnom istragom nije pronađena niti jedna neispravna veza tehnike BuCo Regex.

Analiza 3 pokazuje da tehnika BuCo Regex postiže bolje rezultate nego alat ReLink. U odnosu na etalon, BuCo Regex pronalazi jednak broj neispravnosti BN_i te ima jednaku stopu

Tablica 4.3: Rezultati povezivanja za (0) etalon, (1) ReLink i (2) BuCo Regex

Analiza	**Ulazni podaci**			**Tehnika Povezivanja**	**Izlazni podaci**				
	Izvor	BP_u	BN_u		BV_i	BP_i	BD_i	BN_i	*SP*
2	SVN + Bugs.txt	43,867	673	**(1)**	1014	957	1061	673	100%
	GIT + Bugs.txt	26,287	673	**(2)**	703	664	495	621	93%
3	SVN + Bugs.txt	847	100	**(0)**	127	125	141	81	81%
	SVN + Bugs.txt	847	100	**(1)**	115	113	132	76	76%
	GIT + Bugs.txt	847	100	**(2)**	128	126	141	81	81%

Tablica 4.4: Neispravna veza tehnike BuCo Regex

ID	Opis predaje	Datum izmjene	Datum predaje	Zadužena osoba	Ime programera
9	OPENNLP-190 Updated to Apache **9** parent pom and removed special version which we needed for the Apache 8 parent pom, namely for the rat plugin and the release plugin.	13.1.2011	30.5.2011	William Colen	Joern Kottmann

povezivanja *SP* kao i etalon, dok ReLink pronalazi manji broj neispravnosti BN_i te ima manju stopu povezivanja *SP*. Interesantno je uočiti da čak ni etalon nema stopu povezivanja 100%. Ručna istraga potvrdila je da BuCo Regex uspijeva povezati svih 81 neispravnosti koje etalon povezuje, a pritom čini samo jednu pogrešku. Veza tehnike Buco Regex, koja se nije nalazila u etalonu, prikazana tablicom 4.4, potvrđuje da je riječ o neispravno otkrivenoj vezi. Jednoznamenkasta ID oznaka neispravnosti 9 neispravno je povezana s oznakom inačice projekta Apache 9. Pozitivno je da nije neispravno povezan s ID oznakom neispravnosti 190, koja se također nalazi u opisu predaje, kao što bi to učinila jednostavna pretraga. Potvrda neispravnosti je uočena i u velikoj razlici između datuma izmjene statusa neispravnosti i datuma predaje, te u razlici između imena osobe zadužene za neispravnost i imena programera. Ovaj primjer otkriva nedostatak tehnike BuCo Regex, a to je mogućnost stvaranja neispravnih veza u slučaju da ID oznaka neispravnosti sadrži mali broj znamenaka. Tada se broj ID oznake neispravnosti može podudarati s brojevima unutar drugih ID oznaka, datuma i slično.

Ručna istraga rezultata alata ReLink otkrila je da nema neispravnih veza. Međutim, čak 12 veza nije pronađeno, što čini oko 10% ukupnog broja veza BV_i. Posljedično, 5 neispravnosti ostaje nepovezano, što umanjuje stopu povezivanja *SP* za 5%. Nekoliko primjera veza koje ReLink nije uspio pronaći prikazano je tablicom 4.5. Svi prikazani parametri ukazuju da bi se veze trebale jednostavno pronaći. ID neispravnosti navodi se unutar opisa predaje, datum izmjene statusa neispravnosti jednak je ili blizak datumu predaje, a podudaraju se i ime osobe zadužena za neispravnost i ime programera koji je učinio predaju. Ostaje nejasno zašto ReLink nije uspio pronaći te veze, ali to uzrokuje ozbiljnu sumnju na njegovu učinkovitost. Kada bi se promatrao utjecaj ovih tehnika na konačni skup podataka za SDP, BuCo Regex bi dala identičan skup kao i etalon, a ReLink bi netočno označio 9 datoteka kao da nemaju neispravnosti.

Tablica 4.5: Primjeri nedostajućih veza alata ReLink

ID	Opis predaje	Datum izmjene	Datum predaje	Zadužena osoba	Ime programera
84	OPENNLP-**84** Corrected method name to sentPosDetect	25.1.2011	**25.1.2011**	Joern Kottmann	**joern**
115	OPENNLP-**115** Charset should be specified before creating input stream	11.7.2011	**11.7.2011**	Joern Kottmann	**joern**
471	OPENNLP-**471**: found after we find a name match, we don't jump over the found name but re-process... thanks William for pointing this out	24.4.2012	19.3.2012	James Kosin	**jkosin**

4.3 Diskusija rezultata

U ovom poglavlju predstavljeni su alat BuCo i nova tehnika za povezivanje neispravnosti i predaja programskog koda. Alat BuCo zadovoljava sve parametre prikupljanja podataka koji su otkriveni u pregledu srodne literature i u primjeni, odnosno parametre koji su predstavljeni u prethodnom poglavlju. Razvijeni alat ujedno je i dokaz da je, uz precizno definirane parametre, moguće automatizirano prikupljanje podataka. To može uvelike olakšati buduća istraživanja, ali i pospješiti primjenu u širem kontekstu. Nova tehnika za povezivanje neispravnosti i predaja programskog koda razvijena je na temelju nedostataka najčešće korištene tehnike jednostavne pretrage i napredne i računski zahtjevne kombinacije tehnika koje su implementirane u alat ReLink. Provedenom analizom slučaja pokazano je kako nova tehnika, temeljena na regularnim izrazima, nadmašuje navedene tehnike. Naročito obećavajući rezultati za novu tehniku dobiveni su uporabom etalona skupova podataka, dok je alat ReLink postigao neobjašnjivo loše rezultate u istim uvjetima. Ova analiza slučaja također predstavlja uvod u opsežnu komparativnu studiju, koja je provedena i prikazana u idućem poglavlju.

Valjanost istraživanja

Ovo poglavlje komentira valjanost provedenog istraživanja sukladno preporukama iz poglavlja 1.3. Valjanost zaključaka provedene analize slučaja, koja se prvenstveno odnosi na ponovljivost provedenog istraživanja, nije upitna. Korišteni su razvojni repozitoriji javno dostupnih projekata otvorenog koda te javno dostupni etalon skupova podataka i alat ReLink. S druge strane,

alat BuCo razvijen je sukladno detaljno predstavljenoj proceduri prikupljanja podataka koja definira sve parametre. Valjanost konstrukcije, odnosno prikladnost podataka za ciljeve koji su postavljeni ovoj analizi slučaja, također nije upitna. Međutim, važno je upozoriti na valjanost konstrukcije za istraživanja koja koriste ovako dobivene podatke u daljnjim analizama, kao što je SDP. Čak i u etalonu skupova podataka stopa povezivanja neispravnosti nije 100%, što ukazuje da neke neispravnosti nisu uspješno povezane s programskim kodom. Ovaj podatak bi stoga morao biti naveden, kako bi se moglo bolje procijeniti rizik donošenja krivih zaključaka na temelju tako dobivenih podataka. Vanjska valjanost provedene analize slučaja, koja se prvenstveno odnosi na poopćenje rezultata, ograničena je na analizirane projekte otvorenog koda i etalon skupova podataka. U idućem poglavlju je, stoga, predstavljena opsežna komparativna studija koja će unaprijediti vanjsku valjanost.

Poglavlje 5

Usporedba procedura prikupljanja podataka

Cilj je ove disertacije uspostaviti sustavno definirane smjernice za prikupljanje podataka. Do sada je predstavljeno aktualno stanje u istraživačkoj zajednici, otkrivene su nedosljednosti popularnih pristupa te parametri prikupljanja podataka koji mogu dovesti do odstupanja u prikupljenim podacima ukoliko ih se ostavi otvorenima za interpretaciju [33]. Temeljem svih navedenih spoznaja definirana je procedura, koja obuhvaća sve parametre prikupljanja podataka i razvijena je nova tehnika povezivanja, koja se u provedenoj analizi slučaja pokazala kao obećavajuća [35]. Sve navedeno implementirano je u alatu BuCo, čija namjena je automatizirano prikupljanje podataka. Posljednji korak u uspostavi sustavnih smjernica za prikupljanje je analiza slučaja, čija je svrha objašnjenje uzroka odstupanja u postojećim procedurama i tehnikama prikupljanja podataka. Ova analiza slučaja temelji se na usporedbi najčešće korištenih procedura prikupljanja podataka i tehnika povezivanja razvojnih repozitorija pa se u nastavku disertacije naziva komparativna studija. Usporedba se provodi u razvojnom okruženju svih projekata nad kojima su navedene procedure i tehnike već primijenjene i za koje se nude javno dostupni podaci. Pronalazak tehnike povezivanja koja će davati podatke najveće točnosti je velik izazov. Međutim, to je ujedno i nužno, ukoliko se želi postići usporedivost istraživanja i dobivenih zaključaka. Okvir za provedbu usporedbe tehnika povezivanja razvojnih repozitorija i procedura prikupljanja podataka općenito ne postoji te ga je potrebno uspostaviti. Istraživačka pitanja koja su pri tome postavljena su:

1. Koliki je utjecaj razvojnog okruženja na učinkovitost tehnika povezivanja i zašto?
2. Koja tehnika povezivanja pronalazi najveći broj točnih veza i zašto?

Pojam **razvojnog okruženja** koristi se za općeniti opis karakteristika sadržanih u razvojnom procesu programskog sustava. Razvojno okruženje podrazumijeva strukturu razvojnih repozitorija, posebnosti razvojnog procesa, navike programera te procedure i pravila kojih se pridržavaju. Na primjer, navike programera obično se više razlikuju između programera koji pripadaju različitim razvojnim zajednicama. Međutim, razlike postoje i između programera koji rade na inačicama različitih projekata i između programera koji rade na inačicama istog projekta. Važnu ulogu pri tome imaju konvencije nazivlja, pravila pisanja programskog koda ili druge procedure, koje ne moraju biti opće prihvaćene, već mogu biti posljedica iskustva jedne uže zajednice. Primjerice, razvojna zajednica Eclipse ima vlastitu konvenciju nazivlja, koja može imati značajan utjecaj na kvalitetu podataka ili na odabir prigodne tehnike povezivanja podataka. U ovoj komparativnoj studiji istražuju se razlike između razvojnih zajednica, unutar iste razvojne zajednice i unutar inačica istog projekta. Kako definirati okruženje težak je zadatak i izazov cijele istraživačke zajednice [21]. U ovoj komparativnoj studiji svaki se skup podataka, koji predstavlja jednu inačicu nekog projekta, smatra odvojenim razvojnim okruženjem.

5.1 Metodologija komparativne studije

5.1.1 Materijal komparativne studije

Komparativna studija prati preporuke za provedbu analize slučaja [28]. Višestrukost podataka osigurana je uporabom nekoliko projekata iz dvije zajednice projekata otvorenog koda, Eclipse i Apache. U odabrane projekte već je uložen značajan napor empirijskih istraživanja te stoga pružaju dobru osnovu za usporedbu. Gotovi skupovi podataka iz repozitorija *Eclipse Bug Data*[*] i *Bug prediction dataset*[†] korišteni su u mnogim studijama te su objekt istraživanja i ove studije. Podaci unutar repozitorija *Eclipse Bug Data* prikupljeni su pristupom SZZ pa se istoimeni akronim koristi i za gotove skupove podataka koji su preuzeti iz tog repozitorija. Podaci unutar repozitorija *Bug prediction dataset* su prikupljeni pristupom SZZ koji su unaprijedili D'Ambros i grupa autora pa se prezime vodećeg istraživača (D'Ambros) koristi za skup podataka iz toga repozitorija. Pored spomenutih izvora, objekti ove studije su i već uporabljeni podaci iz alata ReLink i *etalona* skupova podataka. Na navedenim podacima istražena je učinkovitost raznih tehnika povezivanja repozitorija predstavljenih u poglavlju 2.1.1 i pristupa SZZ, odnosno

[*] http://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/
[†] http://bug.inf.usi.ch/

Tablica 5.1: Objekti komparativne studije

Skupovi podataka		
Naziv	Projekt	Sadržaj
SZZ	Eclipse 2.0, 2.1, 3.0	Broj neispravnosti po datoteci
D'Ambros	Eclipse JDT Core 3.4	Broj neispravnosti po razredu
ReLink	Apache HTTPD	Popis veza neispravnosti i predaja
Etalon	Apache OpenNLP	Popis veza neispravnosti i predaja
Alati		
Naziv	Ulazni podaci	Tehnika povezivanja
ReLink	Bugs, SVN	Kombinacija tehnika i predviđanje obradom prirodnog jezika
BuCo Analyzer	Bugzilla GIT	Jednostavna pretraga Vremenska korelacija Podudaranje autorstva Pretraga regularnim izrazom (BuCo Regex)

D'Ambros. Detalji o skupovima podataka i alatima korištenim za primjenu tehnika povezivanja prikazani su u tablici 5.1. Razlika u sadržaju skupova podataka je vrlo važan podatak koji utječe na odabir mjera za predstojeće analize. Skupovi SZZ i D'Ambros sadrže konačni broj neispravnosti po datoteci, ali nisu poznate konkretne veze neispravnosti i datoteka iz kojih taj broj proizlazi. Skupovi alata ReLink i *etalona* sadrže veze neispravnosti i datoteka.

Cilj je pronaći općenito najuspješniju proceduru prikupljanja podataka. Stoga, usporedba tehnika povezivanja ne bi se trebala temeljiti na malom broju skupova podataka koji su javno dostupni. Ova komparativna studija je zato uključila svih 13 inačica projekata Eclipse JDT i PDE, 9 inačica Eclipse projekta BIRT te po jednu inačicu projekta Apache HTTPD i OpenNLP. Veličina tih projekata, izražena brojem inačica, datoteka i neispravnosti te opis njihove domene primjene prikazani su u tablici 5.2.

Za projekte razvojne zajednice Eclipse broj inačica predstavlja broj velikih izdanja koja izlaze jednom godišnje i implementiraju zahtjeve za nove funkcionalnosti. Broj datoteka predstavlja ukupan broj programskih datoteka s ekstenzijom *.java*, a broj neispravnosti predstavlja samo one ispravljene koje su unosile gubitak funkcionalnosti, odnosno one koje će ući u predstojeće analize. Analogno postupku koji koriste SZZ i D'Ambros, kasnije će se ukloniti datoteke za test ili primjer, odnosno one koje ne predstavljaju implementaciju funkcionalnosti u analizi-

Tablica 5.2: Veličina i domena odabranih projekata

Projekti:	**JDT**	**PDE**	**BIRT**	**HTTPD**	**OpenNLP**
Inačica :	13	13	9	1	1
Datoteka:	18,752	6,829	8,104	3,744	1,784
Neispravnosti:	198,206	42,582	65,173	673	100
Domena:	Razvojni alati	Razvojna okolina	Poslovna inteligencija	Mrežni server	Obrada prirodnog jezika

ranim projektima. Takve datoteke prepoznaju se po tome što sadrže *.test* ili *.example*[‡]u nazivu svoje putanje. Za projekte zajednice Apache brojane su datoteke koje imaju *.c* i *.h* ekstenziju te neispravnosti koje daju ReLink i etalon. Kao i za projekte zajednice Eclipse, uklonjene su datoteke koje sadrže *.test* ili *.example* u svojim putanjama.

5.1.2 Varijable i mjere vrednovanja

Uobičajena mjera vrednovanja koja se koristi za usporedbu tehnika povezivanja je stopa povezivanja (*SP*), a ponekad joj se pridodaju i metrike temeljene na tablici zabune [83, 92, 93]. Stopa povezivanja je udio neispravnosti koje su povezane s barem jednom predajom, kao što je spomenuto u poglavlju 2.1.1. Tablica zabune predstavlja sve moguće ishode točnog i netočnog svrstavanja nekih slučaja u klase. Vrednovanje temeljeno na tablici zabune koristi se u domeni klasifikacije. U ovoj studiji slučaji su veze, a klase su postojanje (pozitivno, 1) ili odsutnost (negativno, 0) veze između proizvoljne neispravnosti i predaje. Tablica zabune raspoznaje 4 moguća ishoda:

- *Točno pozitivno* (eng. True Positive, TP) računa se za svaku pronađenu (1) vezu koja stvarno postoji (1);
- *Netočno pozitivno* (eng. False Positive, FP) računa se za pronađene (1) veze koje zapravo ne postoje (0);
- *Netočno negativno* (eng. False Negative, FN) računa se za one veze koje nisu pronađene (0), a zapravo postoje (1);
- *Točno negativno* (eng. True Negative, TN) računa se za one veze koje nisu pronađene (0) i stvarno ne postoje (0).

[‡]*http://wiki.eclipse.org/Naming_Conventions*

Veze koje nedostaju (FN) prepoznate su kao problem u zajednicama projekata otvorenog koda, a manifestiraju se u smanjenoj stopi povezivanja [92, 93]. Zadaća uspješne tehnike povezivanja je imati što veći TP, izbjegavati FP te imati minimalni FN. Kategoriju TN nije jednostavno za razumjeti jer označava sve veze koje neka tehnika nije pronašla, a u stvarnosti niti ne postoje. Ona bi mogla obuhvatiti gotovo sve parove neispravnosti i predaja iz oba repozitorija, što predstavlja jako velik broj pretežno beskorisnih informacija. Osim u slučaju etalona skupova podataka, stvarne veze su a priori nepoznate i zahtijevaju iscrpnu ručnu istragu kako bi se ustanovile. Iz tog razloga, ova studija računa ishode tablice zabune samo za etalon skupa podataka i za prvu inačicu projekta Eclipse JDT, za koju je provedena iscrpna ručna istraga. Za preostale projekte, ručna istraga je provedena na manjem uzorku, pa je utjecaj pojedinih tehnika na ishode tablice zabune samo komentiran. Korištene su sljedeće mjere vrednovanja, izvedene iz tablice zabune: *preciznost* (eng. precision), *osjetljivost* (eng. sensitivity) i *F-mjera* (eng. F-measure). *Točnost* (eng. accuracy) neće biti izračunata, iako je uobičajeno najkorištenija. Ona predstavlja udio točnih ishoda u ukupnom broju ishoda, što uključuje i TN, koji se ne računa.

Mjere vrednovanja uporabljene za sustavnu usporedbu procedura prikupljanja podataka predstavljene su tablicom 5.3. S obzirom na njihovu namjenu, razlikuju se dvije skupine mjera vrednovanja. Jedna skupina mjera koristi se za vrednovanje podataka dobivenih **povezivanjem** neispravnosti i predaja (ReLink i etalon), a druga za vrednovanje podataka dobivenih iz **gotovog skupa** podataka za SDP (SZZ i D'Ambros). Povezivanje otkriva broj veza i njihovih članova te stopu povezivanja, a gotovi skupovi podataka prikrivaju konkretne veze neispravnosti i predaja jer sadrže samo broj neispravnosti za svaku datoteku. Mjere za usporedbu neobrađenih ulaznih podataka zajedničke su za obje skupine. Brojem datoteka (BD_u) i brojem neispravnosti (BN_u) provjerava se imaju li tehnike i pristupi koje uspoređujemo, jednake početne uvjete.

Vrednovanje tehnike povezivanja repozitorija

Primjenom tehnike za **povezivanje** može se ustanoviti broj uspostavljenih veza (BV_i), povezanih predaja (BP_i), povezanih neispravnosti (BN_i) te stopa povezivanja (SP). Usporedba tih brojeva samo površno otkriva u kojoj se mjeri podudaraju dvije tehnike povezivanja. Potrebna je dublja analiza da se ustanovi jesu li otkrivene iste veze te ručna istraga da se ustanovi njihova ispravnost. Stoga, računa se broj podudarajućih veza ($Jednak\ BV_i$), broj ispravnih veza ($Ispravan\ BV_i$) i broj neispravnih veza ($Neispravan\ BV_i$). Analiza slučaja koja je obrađena u poglavlju 4.2 ukazala je na potrebu uvođenja dodatne kategorije veza dvojbene ispravnosti ($Upitan\ BV_i$) kod onih

Tablica 5.3: Mjere vrednovanja u prikupljanju podataka

Metrika	Mjera	Oznaka
Za usporedbu neobrađenih ulaznih i izlaznih podataka		
Ulaznih	Broj datoteka u projektu	(BD_u)
podataka	Broj neispravnosti u projektu	(BN_u)
Izlaznih	Broj uspostavljenih veza	(BV_i)
podataka	Broj povezanih predaja	(BP_i)
nakon	Broj povezanih neispravnosti	(BN_i)
povezivanja	Stopa povezivanja	($SP = BN_i/BN_u$)
SDP skupa	Broj povezanih datoteka	(BD_i)
podataka	Suma neispravnosti u datotekama	(SN_i)
Za usporedbu izlaznih podataka nakon povezivanja		
Sastav	Broj podudarajućih veza	($Jednak\ BV_i$)
izlaznih	Broj ispravnih veza (TP)	($Tocan\ BV_i$)
podataka	Broj neispravnih veza (FP)	($Netocan\ BV_i$)
povezivanja	Broj veza dvojbene ispravnosti	($Upitan\ BV_i$)
Preciznost	Udio točnih veza	($Pr = TP/TP+FP$)
Osjetljivost	Udio pronađenih veza	($Os = TP/TP+FN$)
F-mjera	Srednja vrijednost preciznosti i osjetljivosti	($FM = 2 \cdot Pr \cdot Os/(Pr+Os)$)
Za usporedbu konačnih skupova podataka za SDP		
Sastav 2	Broj datoteka sa jednakim brojem neispravnosti	($Jednak\ BN_i$)
SDP skupa	Broj datoteka sa većim brojem neispravnosti	($Veći\ BN_i$)
podataka	Broj datoteka sa manjim brojem neispravnosti	($Manji\ BN_i$)

veza za koje ne postoje jasne indikacije jesu li ispravne ili ne. Nakon ručne provjere ispravnosti računa se ranije spomenute mjere preciznosti, osjetljivosti i F-mjere. Navedene mjere koriste se u usporedbi tehnika implementiranih BuCo alatom (koje uključuju i primjenu alata ReLink) nad podacima projekata Apache HTTPD (iz alata Relink), Apache OpenNLP (iz etalona) i Eclipse JDT 2.0. Podaci iz alata ReLink i etalona analizirani su u poglavlju 4.2, a JDT 2.0 je predstavnik velikog projekta nad kojim je provedena iscrpna ručna istraga. Budući da za podatke iz projekata Apache HTTPD i JDT 2.0 nemamo uvid u sve TP veze, polazi se od pretpostavke da je taj broj jednak sumi svih ispravno pronađenih veza.

Vrednovanje procedura prikupljanja podataka

Iz **gotovog skupa** podataka može se analizirati broj datoteka koji ima barem jednu neispravnost (BD_i), odnosno barem jednu ostvarenu vezu te suma broja neispravnosti u svim datotekama (SN_i) . Usporedba tih brojeva otkriva u kojoj se mjeri podudaraju konačni produkti prikupljanja podataka, ali ne i jesu li pri tome otkrivene iste veze. Prilikom usporedbe dva gotova skupa, primjenom dublje analize računa se broj datoteka s jednakim (*Jednak* BN_i) i broj datoteka s različitim brojem neispravnosti (*Veći* BN_i i *Manji* BN_i). Dvije kategorije datoteka s različitim brojem neispravnosti uvedene su kako bi olakšale pronalazak procedure koja koristi uspješniju tehniku povezivanja. Ručna istraga se potom usredotočuje na podskup nasumično odabranih datoteka s jednakim i nejednakim brojem neispravnosti. Nastoji se ustanoviti koje su veze dovele do konkretnog broja neispravnosti i koji su razlozi odstupanja. Navedene mjere koriste se u usporedbi pristupa SZZ i D'Ambros, odnosno njihovih skupova podataka i alata BuCo.

5.1.3 Dizajn analize rezultata

Ova komparativna studija temelji se na provedbi metoda kvantitativne analize i analize uzroka neujednačenosti. Cilj, pristup te ulazni i izlazni podaci za svaku metodu prikazani su tablicom 5.4. Sve analize imaju zajednički cilj, a to je otkriti odstupanja u postojećim procedurama prikupljanja podataka za SDP kako bi se razvio točan algoritam. U tablici 5.5 prikazano je koje su tehnike povezivanja repozitorija i pristupi prikupljanja podataka uporabljeni nad kojim od projekata otvorenog koda.

5.2 Rezultati komparativne studije

5.2.1 Kvantitativna analiza

Rezultati prve analize za projekte JDT, PDE i BIRT prikazani su tablicama 5.6, 5.7 i 5.8. Prvi i drugi stupac prikazuju redni broj inačice projekta i broj neispravnosti koji je za njih prikupljen. Preostali stupci prikazuju *SP* za tehnike: jednostavna pretraga, vremenska korelacija (Vrijeme), podudaranje autorstva (Autor), kombinacija tehnika vremenske korelacije i podudaranja autorstva (V & A), BuCo Regex tehnike i alata ReLink. Tehnike vremenske korelacije, podudaranja autorstva i njihova kombinacija su korištene kao unapređenje tehnike jednostavne pretrage jer bi u protivnom pronalazile jako velik broj netočnih (FP) veza. Tehnika vremenske korelacije pove-

Tablica 5.4: Opis analiza korištenih u komparativnoj studiji

Kvantitativna analiza	
Cilj:	procjena odstupanja u prikupljanju podataka
Pristup:	uporaba tehnika povezivanja i skupova podataka iz tablice 5.1
Ulaz:	podaci prikazani tablicom 5.1
Izlaz:	SP, BV_i, BP_i, BD_i, BN_i
Analiza uzroka neujednačenosti	
Cilj:	identifikacija uzroka odstupanja u prikupljanju podataka
Odstupanje tehnika povezivanja	
Pristup:	usporedba BP_i za BuCo, ReLink i etalon
Ulaz:	etalon i veze uspostavljene primjenom BuCo i ReLink alata
Izlaz:	*Jednak* BV_i, *Ispravan* BV_i, *Neispravan* BV_i i *Upitan* BV_i
Odstupanje gotovih skupova podataka	
Pristup:	usporedba BN_i za BuCo, SZZ i D'Ambros pristup
Ulaz:	100 nasumično odabranih datoteka i sve njihove predaje
Izlaz:	*Jednak* BN_i, *Veći* BN_i i *Manji* BN_i

zala bi sve predaje nastale u vrijeme ispravljanja neispravnosti, a tehnika podudaranja autorstva povezala bi sve predaje koje je zadužena osoba ikada učinila. Kada se koriste kao unapređenje tehnike jednostavne pretrage, povezuju predaje koje u svojem opisu imaju i broj ID neispravnosti. Tehnika BuCo Regex nova je tehnika, predstavljena u poglavlju 4.2. ReLink predstavlja kombinaciju tehnika koja je predstavljena u poglavlju 2.1.1. Primjena ReLink alata računski je izuzetno zahtjevna jer gradi i primjenjuje model predviđanja nakon što primijeni kombinaciju jednostavnijih tehnika, pa je on uporabljen samo za projekt JDT. Posljednja dva retka tablica sadrže i srednju vrijednost (μ) i standardnu devijaciju (σ) stope povezivanja, koja je izračunata za svaku tehniku. Na slikama 5.1, 5.2 i 5.3 prikazana je učinkovitost povezivanja primijenjenih tehnika izražena mjerom stope povezivanja *SP* na Y-osi kroz inačice projekata na X-osi.

Usporedba tehnika povezivanja

Analiza rezultata prikazanih tablicama 5.6, 5.7 i 5.8 te slikama 5.1, 5.2, 5.3 dovodi do nekoliko opažanja:

Opažanje 1: alat ReLink povezuje sličan, ali konzistentno veći broj neispravnosti u odnosu

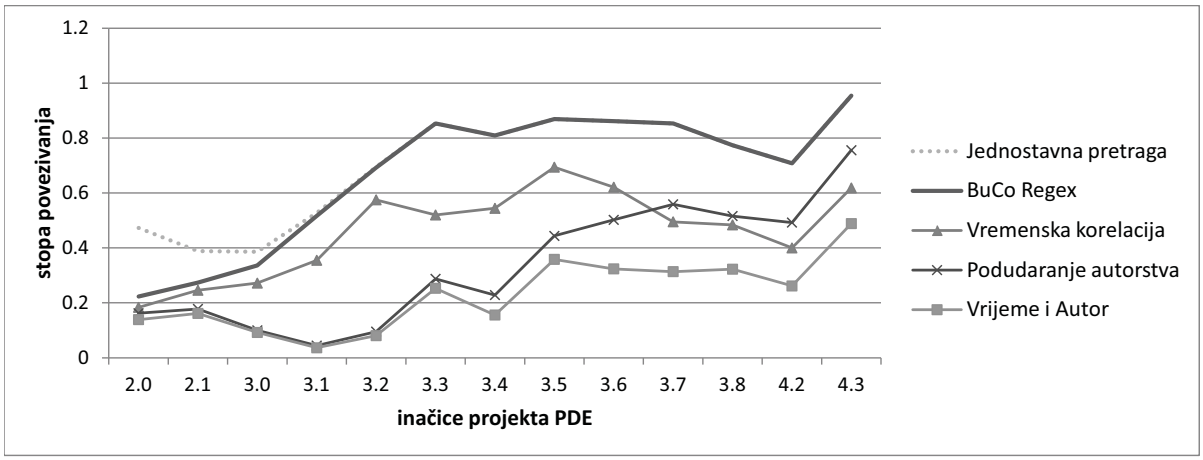
Tablica 5.5: Kombinacije odabranih projekata i korištenih tehnika povezivanja

Projekt otvorenog koda	Primjenjene tehnike povezivanja
Apache OpenNLP	etalon skupa podataka
	ReLink
	Buco Regex
Apache HTTPD	etalon skupa podataka
	ReLink
	Buco Regex
Eclipse JDT 2.0, 2.1, 3.0	ReLink (samo JDT)
	pristup SZZ
	Buco Regex
Eclipse PDE 2.0, 2,1, 3.0	jednostavna pretraga
	jednostavna pretraga i vremenska korelacija
	jednostavna pretraga i podudaranje autorstva
	jednostavna pretraga, vremenska korelacija i podudaranje autorstva
Eclipse JDT 3.1 - 4.3	ReLink (samo JDT)
	D'Ambros pristup (samo za JDT Core 3.4)
	Buco Regex
Eclipse PDE 3.1 - 4.3	jednostavna pretraga
	jednostavna pretraga i vremenska korelacija
	jednostavna pretraga i podudaranje autorstva
	jednostavna pretraga, vremenska korelacija i podudaranje autorstva
Eclipse BIRT 2.0 - 4.3	Buco Regex
	jednostavna pretraga
	jednostavna pretraga i vremenska korelacija
	jednostavna pretraga i podudaranje autorstva
	jednostavna pretraga, vremenska korelacija i podudaranje autorstva

na tehniku BuCo Regex. Razlika između dvije navedene tehnike vrlo je malena, iznosi oko 1%, ali je prisutna u svakoj od 13 analiziranih inačica projekta JDT. Primjena tehnika obrade prirodnog jezika i modela predviđanja, koje alat ReLink koristi za pronalazak veza koje nedostaju, vrlo je vjerojatno uzrok konzistentnoj razlici. Ručna istraga u analizi uzroka neujednačenosti detaljnije će se osvrnuti na ovu pretpostavku.

Tablica 5.6: Usporedba stope povezivanja za tehnike povezivanja u projektu JDT

JDT		Jednostavna	Unapređenje			BuCo	ReLink
Inačica	BN_u	pretraga	Vrijeme	Autor	V & A	Regex	Alat
1	4276	84.5%	32.3%	18.5%	14.3%	48.4%	49.3%
2	1875	73.1%	30.9%	22.9%	16.9%	64.4%	65.3%
3	3385	73.9%	27.5%	20.4%	11.3%	70.9%	71.5%
4	2653	81.6%	25.0%	25.4%	13.5%	80.6%	81.5%
5	1879	85.0%	23.6%	30.5%	13.5%	84.9%	85.8%
6	1341	88.7%	28.7%	35.8%	16.0%	88.7%	89.8%
7	989	90.0%	30.5%	42.2%	19.0%	90.0%	91.5%
8	595	91.8%	45.4%	65.0%	30.9%	91.8%	92.8%
9	492	88.6%	44.1%	75.2%	37.4%	88.6%	90.2%
10	549	72.7%	34.2%	64.8%	30.4%	72.7%	73.8%
11	329	87.5%	54.7%	66.0%	41.0%	87.5%	90.9%
12	41	87.8%	65.9%	73.2%	56.1%	87.8%	90.2%
13	348	89.7%	73.0%	51.1%	43.1%	89.7%	90.8%
μ		84.2%	39.7%	45.5%	26.4%	80.5%	81.4%
σ		6.8%	16.0%	21.3%	14.4%	12.9%	12.9%

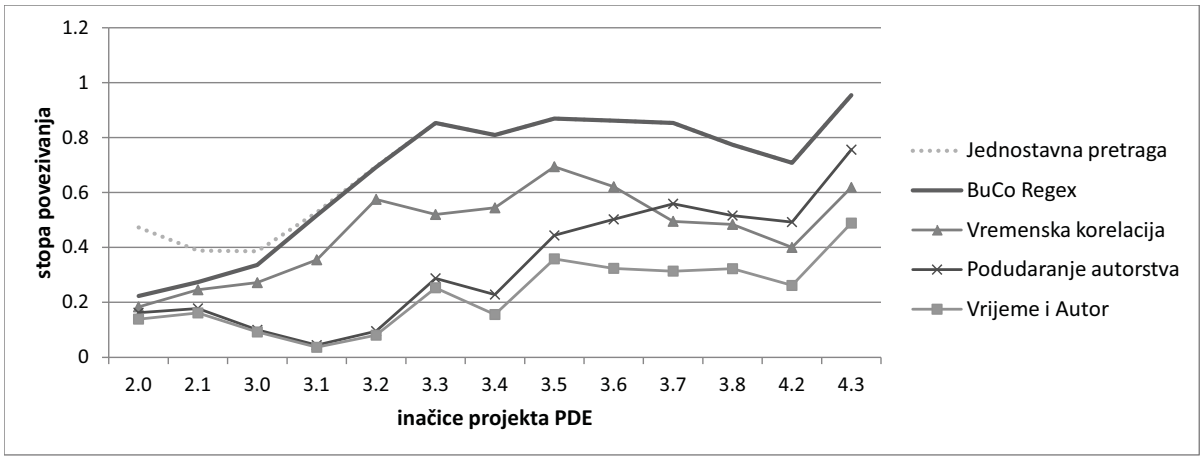


Slika 5.1: Stopa povezivanja za tehnike povezivanja u projektu JDT

Opažanje 2: Tehnike vremenske korelacije i podudaranja autorstva uspostavljaju mnogo manje veza nego druge tehnike. Tehnike vremenske korelacije i podudaranja autorstva osla-

Tablica 5.7: Usporedba stope povezivanja za tehnike povezivanja u projektu PDE

PDE		Jednostavna	Unapređenje			BuCo
Inačica	BN_u	pretraga	Vrijeme	Autor	V & A	Regex
1	561	47.2%	18.4%	16.2%	13.9%	22.3%
2	427	38.9%	24.6%	17.8%	16.2%	27.4%
3	1041	38.6%	27.2%	10.0%	9.2%	33.6%
4	769	52.8%	35.5%	4.4%	3.6%	51.6%
5	546	69.4%	57.5%	9.5%	8.1%	69.2%
6	727	85.3%	52.0%	28.7%	25.3%	85.3%
7	963	80.9%	54.4%	22.8%	15.6%	80.9%
8	879	86.9%	69.4%	44.4%	35.8%	86.9%
9	454	86.1%	62.1%	50.2%	32.4%	86.1%
10	204	85.3%	49.5%	55.9%	31.4%	85.3%
11	62	77.4%	48.4%	51.6%	32.3%	77.4%
12	65	70.8%	40.0%	49.2%	26.2%	70.8%
13	131	95.4%	61.8%	75.6%	48.9%	95.4%
μ		70.4%	46.2%	33.6%	23.0%	67.1%
σ		19.6%	15.9%	22.2%	13.1%	25.0%

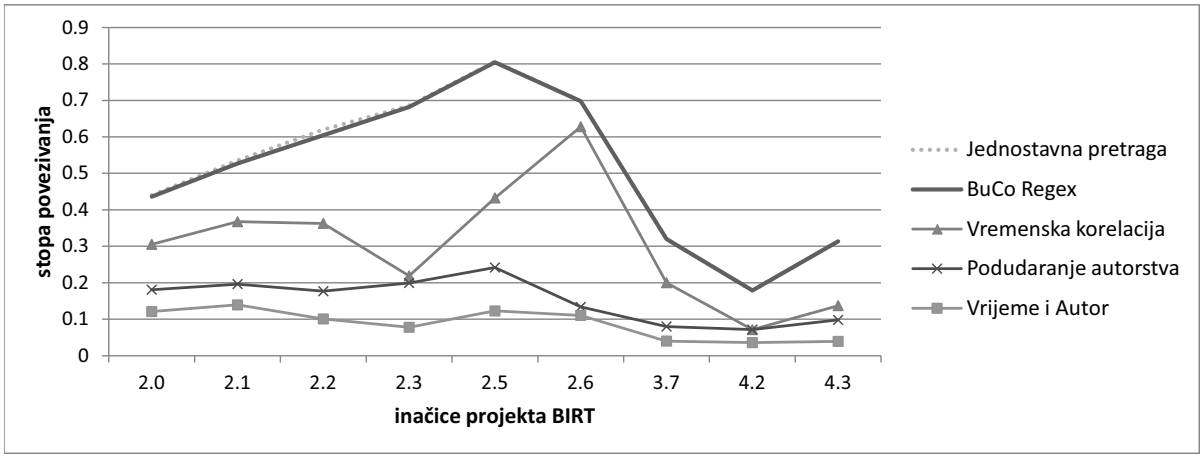


Slika 5.2: Stopa povezivanja za tehnike povezivanja u projektu PDE

njaju se na informacije unesene uz predaje i neispravnosti koje nisu formalno definirane niti propisane pravilima razvojnih zajednica. One su prepoznate kao jedan od mogućih temelja za

Tablica 5.8: Usporedba stope povezivanja za tehnike povezivanja u projektu BIRT

BIRT		Jednostavna	Unapređenje			BuCo
Inačica	BN_u	pretraga	Vrijeme	Autor	V & A	Regex
1	2034	44.0%	30.5%	18.1%	12.1%	43.6%
2	596	53.5%	36.7%	19.6%	13.9%	52.7%
3	2630	61.9%	36.3%	17.7%	10.1%	60.5%
4	1761	68.5%	21.9%	19.9%	7.8%	68.2%
5	807	80.5%	43.2%	24.2%	12.3%	80.4%
6	172	69.8%	62.8%	13.4%	11.0%	69.8%
7	25	32.0%	20.0%	8.0%	4.0%	32.0%
8	28	17.9%	7.1%	7.1%	3.6%	17.9%
9	51	31.4%	13.7%	9.8%	3.9%	31.4%
μ		51.1%	30.3%	15.3%	8.7%	50.7%
σ		21.1%	16.9%	6.0%	4.0%	20.9%



Slika 5.3: Stopa povezivanja za tehnike povezivanja u projektu BIRT

povezivanje, ali očito mogu unijeti veliko odstupanje. U usporedbi s tehnikom BuCo Regex, tehnika vremenske korelacije ima *SP* manju za faktor u rasponu 1.2 - 3.6 za JDT, 1.1 - 1.7 za PDE i 1.1 - 3.2 za BIRT; tehnika podudaranje autorstva ima *SP* manju za faktor u rasponu 1.1 - 3.4 za JDT, 1.2 - 11.6 za PDE i 2.4 - 5.2 za BIRT; kombinacija tehnika vremenske korelacije i podudaranja autorstva ima *SP* manju za faktor u rasponu 1.5 - 6.3 za JDT, 1.6 - 14.1 za PDE i 3.6 - 8.7 za BIRT. Ove tehnike imaju mnogo veza koje nedostaju (FN) te se, posljedično, ne

mogu koristiti za povezivanje. Njihova primarna uloga trebala bi biti provjera ispravnosti veza upitne ispravnosti.

Opažanje 3: Tehnike BuCo Regex i jednostavna pretraga imaju jednak SP nakon nekoliko inačica projekta. Nakon 5. inačice projekata PDE i BIRT, odnosno nakon 6. inačice projekta JDT, ove tehnike povezuju jednak broj neispravnosti. Razlika između njih je u tome što tehnika BuCo Regex definira koji znakovi smiju okružiti ID broj neispravnosti u opisu izmjene. Ovim unapređenjem postiže se točnije povezivanje, što se naročito manifestira u ranijim inačicama. Tada brojevi unutar ID oznaka imaju manje znamenaka, zbog čega ih jednostavna pretraga može neispravno pronaći u sklopu drugih ID oznaka ili općenito drugih brojeva. U kasnijim inačicama projekata brojevi ID oznaka neispravnosti postaju veći i tada su ove tehnike jednako učinkovite.

Opažanje 4: Stopa povezivanja raste u kasnijim inačicama. Kod svih projekata uočen je porast stope povezivanja u kasnijim inačicama. Rastući trend stope povezivanja najizraženiji je u najučinkovitijim tehnikama, BuCo Regex i ReLink. Najvjerojatniji razlog je što razvojna zajednica, njezini programeri i razvojni principi evoluiraju zajedno s projektom. S vremenom postaju konzistentniji u prijavljivanju ID brojeva neispravnosti unutar opisa predaja. Najučinkovitijim tehnikama *SP* raste od približno 50% do čak 90% za JDT, od 22% do 80% za PDE i od 43% do 80% za BIRT. Iz toga se može zaključiti da su kasnije inačice projekata manje sklone odstupanjima unesenima od strane programera te da su podaci više kvalitete. Tehnike vremenske korelacije i podudaranja autorstva također imaju rastući trend, ali njihove stope povezivanja su i dalje mnogo manje od ostalih tehnika.

Opažanje 5: Projekti sličnijeg razvojnog okruženja imaju sličniji trend stope povezivanja. Projekti JDT i PDE imaju paralelni razvoj, u inačicama jednakih oznaka i datuma izdavanja. Za te projekte uočena je sličnost trenda *SP*-a, koja raste iznad 80%, pa čak do 95%, te ostaje na visokoj razini i u najnovijim inačicama. S druge strane, projekt BIRT ima nešto drugačiji tijek razvoja od projekata JDT i PDE. Za njegove inačice trend *SP*-a je drugačiji. U samo jednoj inačici prelazi 80%, a u posljednjim inačicama opada na 33%. To ukazuje da različito razvojno okruženje može biti prisutno čak i unutar iste razvojne zajednice te da može imati različit utjecaj na odstupanje povezivanja.

Utjecaj razvojnog okruženja na prikupljanje

Tablica 5.9 predstavlja neobrađene ulazne i izlazne podatke nakon primjene procedura prikupljanja i tehnika povezivanja u okruženju različitih projekata i njihovih inačica. Mjere ulaznih i izlaznih podataka BD_u, BD_i i SN_i izračunate su samo za datoteke programskog koda (ekstenzije *.c* i *.h* za programski jezik C++, odnosno ekstenzije *.java* za programski jezik Java). Jednaki BD_i pokazuje da su osigurani identični početni uvjeti za prikupljanje te dobra osnova za usporedbu. Jedina iznimka je projekt Apache HTTPD za koji ReLink koristi sustav SVN, a BuCo koristi sustav GIT. Budući da skupovi podataka SZZ i D'Ambros daju samo konačni broj neispravnosti po datotekama, nije poznat točan BN_u, BV_i i BP_i. Redoslijed pretrage njihove procedure počinje od repozitorija sustava za upravljanje inačicama, što se pokazalo kao manje uspješan redoslijed od onoga koji koristi alat BuCo, te predstavlja prepreku u rekonstrukciji njihovog postupka. Primjenom određene tehnike povezivanja razvojnih repozitorija dobiju se veze neispravnosti i predaja. Stoga, alat BuCo provodi preslikavanje tih veza s predaja na datoteke, te se računa BD_i i SN_i. Važno je naglasiti da SN_i i SP nisu linearno korelirani jer jedna neispravnost može biti (i često jest) prisutna u većem broju datoteka. Analiza rezultata predstavljenih u tablici 5.9 dovodi do sljedećeg opažanja:

Opažanje 6: Učinkovitost tehnike povezivanja ovisi o razvojnom okruženju. Tehnika BuCo Regex povezuje mnogo veći broj datoteka (BD_i) nego procedura SZZ. Ta razlika varira između 77% i 85% za projekt PDE. S druge strane, razlika povezanih datoteka je mnogo manja za projekt JDT, gdje u prve 2 inačice povezuje manje, a u 3. inačici neznatno više datoteka. Razlika varira između 1% i 6.5% u usporedbi sa SZZ i 33% u usporedbi s procedurom D'Ambros. Alat ReLink pronalazi više veza od tehnike BuCo Regex u svim slučajevima, osim za projekt Apache OpenNLP, gdje povezuje otprilike 20% manje ND_i i SN_i. Utjecaj razvojnog okruženja može se primijetiti na 3 razine:

- *Razina razvojne zajednice* - usporedba između alata ReLink i tehnike BuCo Regex, za projekte iz različitih razvojnih zajednica otkriva da ReLink propušta veze mnogo više za projekt Apache HTTPD nego za ostale projekte;
- *Razina projekta* - usporedba između tehnike BuCo Regex i pristupa SZZ za projekte iz iste razvojne zajednice otkriva da BuCo Regex povezuje mnogo više datoteka za PDE, a manje datoteka za JDT projekt;
- *Razina inačice* - usporedba mjere BD_i između tehnike BuCo Regex i pristupa SZZ i D'Ambros za inačice istog JDT projekta otkriva da BuCo Regex povezuje manje u pr-

Tablica 5.9: Rezultat primjene procedura prikupljanja i tehnika povezivanja

Ulazni podaci			**Postupak**	**Izlazni podaci**			
Izvor	BD_u	BN_u	**Prikupljanja**	BV_i	BP_i	BD_i	SN_i
	2406	nepoznato	SZZ	nepoznato	nepoznato	1188	3860
JDT 2.0	2406	4276	BuCo Regex	1809	1733	1111	3794
	2406	4276	ReLink	1838	1739	1122	3924
	2748	nepoznato	SZZ	nepoznato	nepoznato	1008	2506
JDT 2.1	2748	1875	BuCo Regex	1085	1066	875	2142
	2748	1875	ReLink	1095	1068	817	2134
	3292	nepoznato	SZZ	nepoznato	nepoznato	1271	3498
JDT 3.0	3292	3385	BuCo Regex	2212	2116	1289	4423
	3292	3385	ReLink	2231	2114	1300	4465
PDE 2.0	567	nepoznato	SZZ	nepoznato	nepoznato	25	34
	567	561	BuCo Regex	119	112	113	246
PDE 2.1	765	nepoznato	SZZ	nepoznato	nepoznato	22	42
	765	427	BuCo Regex	123	122	119	229
PDE 3.0	1041	nepoznato	SZZ	nepoznato	nepoznato	42	77
	1041	1041	BuCo Regex	330	329	282	584
	997	nepoznato	D'Ambros	nepoznato	nepoznato	206	374
JDT Core	997	989	BuCo Regex	855	839	316	719
3.4	997	989	ReLink	883	861	317	820
	1784	100	Etalon	127	125	100	131
OpenNLP	1784	100	BuCo Regex	128	126	100	131
	1784	100	ReLink	115	113	80	102
HTTPD	2243	673	BuCo Regex	703	664	138	624
	3744	673	ReLink	1014	957	280	857

vim inačicama, a više u narednim inačicama.

Naredne analize nastoje rasvijetliti koji čimbenici utječu na ovakve rezultate.

5.2.2 Analiza uzroka neujednačenosti

Prethodne usporedbe temeljila su se na mjerama BV_i, BD_i te SN_i, ali kako te mjere ne pružaju dovoljnu količinu informacija, potrebne su dodatne analize. Detaljna analiza uzroka neujednačenosti započinje **usporedbom tehnika povezivanja**. Provjerava se ispravnost veza koje su pronađene različitim tehnikama povezivanja. Za projekt OpenNLP, veze koje su tehnike pronašle uspoređuju se s vezama iz etalona. Za projekt JDT 2.0, provodi se iscrpna ručna istraga svih 3,647 veza koje su povezali alat ReLink i tehnika BuCo Regex.

Analiza uzroka neujednačenosti potom se provodi **usporedbom konačnih skupova podataka**. Za razliku od usporedbe tehnika povezivanja, u konačnim skupovima podataka, kao što su SZZ i D'Ambros, nisu poznate veze koje su ostvarene prikupljanjem. Ručna istraga je nužna kako bi se ustanovila ispravnost podataka. Uspoređuje se broj neispravnosti BN_i koji sadrže datoteke iz gotovih skupova podataka SZZ i D'Ambros s brojem neispravnosti koji sadrže iste datoteke u skupovima podataka dobivenim alatom BuCo. Računa se broj datoteka koji ima jednak broj neispravnosti ($Jednak\ BN_i$), te broj datoteka za koji BuCo Regex pronalazi više neispravnosti ($Veći\ BN_i$), odnosno manje neispravnosti ($Manji\ BN_i$). Potom se nasumično odabire podskup od 100 datoteka, među kojima su jednako zastupljene datoteke s jednakim i nejednakim brojem neispravnosti. Za odabrane datoteke kompletno se rekonstruira povezivanje koje su provodile procedure SZZ i D'Ambros te se uspoređuju s vezama tehnike BuCo Regex.

Usporedba tehnika povezivanja

Iscrpna analiza veza koje su pronađene tehnikama BuCo Regex i ReLink provedena je za projekte JDT 2.0 i OpenNLP. Za projekt JDT 2.0 pronađene veze uspoređene su međusobno, a iscrpna ručna istraga bila je nužna radi utvrđivanja njihove ispravnosti, budući da ispravne veze nisu unaprijed poznate. Za projekt OpenNLP pronađene veze uspoređene su s etalonom skupa podataka koji sadrži sve ispravne veze. Rezultati ove analize prikazani su u tablici 5.10. Tablica je vertikalno podijeljena u dva dijela koji sadrže rezultate primjene obje tehnike za projekte JDT 2.0 i OpenNLP. Tablica je horizontalno podijeljena u tri dijela koji sadrže broj pronađenih veza BV_i korištenjem svake tehnike, njihov sastav s obzirom na podudaranje i ispravnost, te preciznost, osjetljivost i F-mjeru. Veze koje su pronađene korištenjem obiju tehnika ($Jednak\ BV_i$) pokazale su se ispravnima, a čine preko 90% veza za obje tehnike. Veze koje je pronašla samo jedna tehnika kategorizirane su s obzirom na ispravnost kao ispravne ($Ispravan\ BV_i$), neispravne ($Neispravan\ BV_i$), te one kojima nije ustanovljena niti opovrgnuta ispravnost ($Upitan\ BV_i$). In-

Tablica 5.10: Detaljna usporeba tehnika za povezivanje

Projekt:	JDT 2.0		OpenNLP	
Tehnika:	Relink	BuCo Regex	Relink	BuCo Regex
BV_i	2979	2830	115	128
Jednak BV_i	2793 93.8%	2793 98.7%	115 90.6%	127 100.0%
Ispravan BV_i	12 0.4%	34 1.2%	0 0.0%	11 8.6%
Neispravan BV_i	61 2.0%	3 0.1%	0 0.0%	1 0.8%
Upitan BV_i	114 3.8%	0 0.0%	0 0.0%	0 0.0%
Preciznost	94.1% - 97.9%	99.9%	100%	99.2%
Osjetljivost	98.8%	99.6%	90.5%	100%
$F-mjera$	96.4% - 98.3%	99.7%	95.1%	99.6%

dikator ispravnosti veze može biti ako ID neispravnosti postoji u opisu predaje, ako je autor predaje osoba zadužena za neispravnost, ili je zadovoljen kriterij podudaranja vremena predaje i zatvaranja neispravnosti. Kod veza dvojbene ispravnosti nije pronađen osnovni kriterij ispravnosti, a to je ID neispravnosti. Pored broja pronađenih veza prikazan je i njihov udio u ukupnom broju svih ispravno pronađenih veza. Preciznost, Osjetljivost i F-mjera računaju se sukladno formulama prikazanima u tablici 5.3. U slučaju projekta JDT 2.0, preciznost i F-mjera sadrže dvije vrijednosti. One predstavljaju najgori i najbolji scenarij, s obzirom na veze dvojbene ispravnosti. Manja vrijednost tih mjera je za slučaj da su veze neispravne (FP), a veća vrijednost je za slučaj da su veze ispravne (TP). Razmatranjem rezultata prikazanih tablicom 5.10, dolazi se do narednih opažanja:

Opažanje 7: Uporaba modela predviđanja kao tehnike povezivanja može biti nadmašena jednostavnijom tehnikom zasnovanom na regularnim izrazima. Alat ReLink i tehnika BuCo Regex ostvaruju preko 90% ispravnih veza, ali sadrže i neke netočne (FP) veze ili im nedostaju neke (FN) veze. Ostaje nejasno zašto alat ReLink nije pronašao 45 veza, odnosno 1.6% TP veza koje pronalazi BuCo Regex. Od 12 veza pronađenih alatom ReLink, odnosno 0.4% koje BuCo Regex ne pronalazi, 8 ih predstavlja duple veze između istih neispravnosti i datoteka, 3 ih predstavlja duple unose u različitim podrepozorijima sustava za upravljanje inačicama, a samo 1 veza predstavlja uspješno pronađenu tipografsku pogrešku. ReLink je uspješno povezao neispravnost 20741 s predajom čiji opis glasi "*Fixed bug 207141: Inexact Matches dialog check box isn't properly positioned [search]*". Neispravnosti 207141 koja je spomenuta unutar opisa

prijavljena je 2007. godine za projekt BIRT. Povezana predaja nastala je 2002. godine, na isti dan kada je neispravnost 20741 zatvorena. Alat ReLink ima za cilj pronaći upravo ovakve tipografske pogreške, ali zbog toga često pronalazi netočne (FP) veze. Za čak 61 vezu, odnosno 2% od ukupnog broja veza, netočno je predviđena tipografska pogreška. Broj ID oznake netočno povezanih neispravnosti razlikuje se u čak dvije znamenke od broja ID oznake koji se nalazi u opisu. U nekim slučajevima, ista predaja je ponekad već bila točno povezana sa neispravnošću odgovarajućeg broja ID oznake. Primjerice, predaja čiji opis glasi "*bug 9456*" točno je povezana s ID-om neispravnosti 9456, ali ReLink je istu predaju povezao i s ID-om 9591 koji nije imao drugih veza. Za sve 4 veze tehnike BuCo Regex koje ReLink nije pronašao, broj ID oznake neispravnosti je sadržavao mali broj znamenaka. Za 3 veze iz projekta JDT 2.0, broj ID oznake koji ima četiri ili manje znamenaka bio je sadržan i u imenu datoteke. Zbog svega navedenog, BuCo Regex tehnika nadmašuje ReLink alat u mjerama preciznosti, osjetljivosti i F-mjere. Jedini izuzetak je preciznost za OpenNLP u kojem BuCo Regex ima jednu FP vezu, ali zato ReLink za isti projekt ne pronalazi čak 10% veza.

Opažanje 8: Stopa povezivanja može dovesti do neispravnog zaključka glede učinkovitosti povezivanja. Viša stopa povezivanja te veći BD_i i SN_i ne moraju nužno značiti i veću učinkovitost povezivanja. ReLink je nadmašio BuCo Regex u navedenim mjerama u *opažanju 4*, ali dublja analiza otkriva suprotno. Veze koje otkriva BuCo Regex su ispravne u 99.2% slučajeva, dok ReLink ima točnost od 94.2%. BuCo Regex otkriva samo 0.4% manje ispravnih veza od ReLink-a, a ReLinku nedostaje 1.2% veza. Od 114 ReLink-ovih veza dvojbene ispravnosti, 76 predaja je nastalo unutar 10 dana od zatvaranja neispravnosti, 25 veza zadovoljava uvjet tehnike podudaranja autorstva, a samo 18 veza zadovoljava oba uvjeta. Bez pomoći programera koji su razvijali taj sustav nije moguće sa sigurnošću utvrditi ispravnost takvih veza.

Usporedba konačnih skupova podataka

Detaljna analiza konačnih skupova podataka koji su proizašli iz procedure SZZ, D'Ambros i alata BuCo Regex provedena je za nekoliko inačica projekata Eclipse JDT i PDE. Konačni skupovi podataka sadrže broj neispravnosti u svakoj datoteci. Prvom analizom uspoređen je broj datoteka za koje su dvije procedure prikupile isti, odnosno različit broj neispravnosti. Rezultati su prikazani tablicom 5.11. Tablica je horizontalno podijeljena na 3 dijela, od kojih prva dva uspoređuju podatke SZZ za prve 3 inačice projekata JDT i PDE, a treći dio uspoređuje podatke D'Ambros za 7. inačicu (3.4) komponente Core iz projekta JDT. Vertikalno su prikazani brojevi

Tablica 5.11: Detaljna usporedba konačnih skupova podataka za SDP

Projekt	Ukupno datoteka	*Jednak* BN_i	*Veći* BN_i	*Manji* BN_i
JDT 2.0	2406	1740 (72.3%)	341 (14.2%)	325 (13.5%)
JDT 2.1	2748	1383 (50.3%)	605 (22.0%)	760 (27.7%)
JDT 3.0	3292	1442 (43.8%)	970 (29.5%)	880 (26.7%)
PDE 2.0	575	482 (83.3%)	90 (15.7%)	3 (0.5%)
PDE 2.1	765	666 (87.1%)	97 (12.7%)	1 (0.1%)
PDE 3.0	874	621 (71.1%)	253 (28.9%)	0 (0.0%)
JDT Core 3.4	997	621 (62.3%)	253 (25.3%)	123 (12.3%)

datoteka za koje BuCo Regex pronalazi *Jednak* BN_i, *Veći* BN_i ili *Manji* BN_i, te njihov udio u ukupnom broju datoteka.

Drugom analizom su traženi uzroci koji su doveli do odstupanja skupova podataka. Za nasumično odabrani podskup datoteka, pronađeni uzroci odstupanja su i kvantificirani. Nasumično je odabrano 100 datoteka, s jednakim 50% : 50% udjelom datoteka koje imaju jednak i onih koje imaju različiti broj neispravnosti, te sa 60% : 40% udjelom iz JDT, odnosno PDE projekta. Ustanovljena su tri uzroka odstupanja u procedurama prikupljanja podataka koji su doveli do različitih BN_i u datotekama. Rezultati ručne istrage prikazani su tablicom 5.12. Tablica je horizontalno podijeljena po projektima, a vertikalno po pronađenim uzrocima odstupanja. Uzroci neujednačenosti poredani su od najzastupljenijeg do najmanje zastupljenog, s lijeva na desno. Detaljniji opisi odstupanja navedeni su u nastavku.

Odstupanje 1 ($\Delta t > 6$ mjeseci): za razliku od procedure alata BuCo, procedure SZZ i D'Ambros prilikom prikupljanja neispravnosti ne uzimaju u obzir oznaku inačice projekta za koju su te neispravnosti prijavljene u sustavu za praćenje zahtjeva. Umjesto toga, te procedure neispravnosti svrstavaju u onu inačicu čiji je datum izdanja unutar 6 mjeseci od datuma prijave neispravnosti. Ovo odstupanje se manifestira kao odstupanje u nekoliko razina. Primjerice, neispravnost 6839 prijavljena je u prosincu 2001. godine, što je više od 6 mjeseci prije datuma izdanja inačice JDT 2.0, za koju je prijavljena u sustavu za praćenje zahtjeva. Neispravnost se povezuje s predajom čiji opis glasi "*Fix for 6839*". Datum predaje je siječanj 2002. godine, 2 dana nakon što je neispravnost zatvorena. Zbog razlike veće od 6 mjeseci, procedura SZZ propušta ovu očitu vezu te posljedično, jedna datoteka ima jednu neispravnost manje u odnosu na alat BuCo. Primjer većeg odstupanja u broju neispravnosti pronađen je za neispravnosti 28622,

28559 i 28682. Ove neispravnosti prijavljene su u prosincu 2002. godine, što je unutar 6 mjeseci od datuma izdanja inačice JDT 2.0. Međutim, neispravnosti su prijavljene za inačicu JDT 2.1. Povezuje ih se sa predajama čiji opisi glase "*Fixes for 28559, 28622 and 28682*" i "*Fix for 28682*". *Odstupanje 1* sada se manifestira tako da 3 datoteke iz skupa SZZ imaju SN_i veći za 7 neispravnosti u inačici JDT 2.0, a za 7 manji u inačici 2.1.

Odstupanje 2 (Nedostaje ključna riječ): Za razliku od procedure alata BuCo, procedure SZZ i D'Ambros povezuju samo predaje koje sadrže neku od ključnih riječi unutar opisa. To može biti riječ "*bug*" ili "*fix*", ili znak "#" ispred ID oznake neispravnosti [65]. Inzistiranje na ključnim riječima je uzrok *odstupanja 2*, a datoteka ClassFileStruct.java[§] je primjer koji to dokazuje. Za ovu datoteku, SZZ skup podataka pronalazi 0 neispravnosti, a alat BuCo ju povezuje s dvije neispravnosti. Obje neispravnosti povezane su sa predajom čiji opis glasi "*Update for 10979 and 10697*". Osim što predaja unutar opisa sadrži ID oznaku neispravnosti, ona je nastala unutar jednog dana od datuma zatvaranja neispravnosti, a autor predaje, Olivier Thomannis, je ujedno i osoba zadužena za neispravnost 10697. Iako je veza višestruko potvrđena, procedura SZZ ju propušta zbog toga što u opisu spomenute predaje nema ključnih riječi. Iz istog razloga nije uspostavljena niti veza s predajom čiji opis glasi "*Update for 10697*". Ovaj primjer *Odstupanja 2* manifestira se tako da skup podataka SZZ ima BD_i manji za 7 datoteka, a SN_i manji za 13 neispravnosti.

Odstupanje 3 (Drugi projekt): Za razliku od procedure alata BuCo, procedure SZZ i D'Ambros uključuju neispravnosti bez obzira za koji projekt su prijavljene. Datoteka ASTNode.java[¶] je primjer koji ukazuje na navedeno odstupanje. Za ovu datoteku, procedura SZZ pronalazi 9 neispravnosti prije izdanja i 1 neispravnost poslije izdanja, dok BuCo pronalazi 8 neispravnosti prije izdanja i 1 neispravnost poslije izdanja. Pretraživši cijeli repozitorij sustava GIT za predajama koje su mijenjale navedenu datoteku, pronađene su dvije potencijalne veze. To su predaje čiji opisi glase: "*Fix for 10495*" i "*Add useful toString methods for ASTNodes (bug 11076)*". Budući da je neispravnost 10495 trivijalne težine, koju ni procedura SZZ ni procedura alata BuCo ne uzimaju u obzir, očito je da neispravnost 11076 čini razliku. Međutim, navedena neispravnost nije prijavljena za projekt Eclipse JDT, već za Eclipse Platform. Ovaj primjer *odstupanja 3* manifestira se tako da procedura SZZ ima SN_i veći za 6 neispravnosti.

Razmatranjem rezultata prikazanih tablicama 5.11 i 5.12, dolazi se do narednih opažanja:

[§]org/eclipse/jdt/internal/compiler/classfmt/ClassFileStruct.java
[¶]org/eclipse/jdt/core/dom/ASTNode.java

Tablica 5.12: Ručna istraga uzroka odstupanja skupova podataka

Projekt	Datoteka	Δ t > 6 mjeseci	Nedostaje ključna riječ	Drugi projekt
JDT	60	16 (26.6%)	9 (15.0%)	3 (5.0%)
PDE	40	13 (32.5%)	8 (20.0%)	1 (2.5%)
Ukupno:	100	29 (29.0%)	17 (17.0%)	4 (4.0%)

Opažanje 9: Učinkovitost povezivanja može biti narušena prestrogo definiranom procedurom. Jedna od preporuka koje su dali Basilli i Weis u metodologiji ispravnog prikupljanja podataka u programskom inženjerstvu govori da forme za prikupljanje moraju pružati određeni stupanj fleksibilnosti, kako se ne bi ispustilo određene podatke [57]. Zahtjevi da neispravnost bude prijavljena unutar 6 mjeseci od datuma izdanja inačice za koju su prijavljeni (*odstupanje 1*), odnosno da opis predaje sadrži neku točno određenu ključnu riječ, (*odstupanje 2*) pokazali su se kao prestrogi zahtjevi. Oni ne pružaju fleksibilnost, uzrok su ispuštanju određenih informacija te narušavaju učinkovitost procedure prikupljanja. *Odstupanje 1* je najveći uzrok odstupanja skupova podataka između procedura SZZ i D'Ambros i alata BuCo. Utječe na 29 od 100 analiziranih datoteka, što je zapravo 58% od datoteka koje imaju različit BN_i. *Odstupanje 2* je drugi najčešći uzrok odstupanja, koji utječe na 17 od 100 datoteka, odnosno na 34% datoteka koje imaju različit BN_i. *Odstupanje 3* je najrjeđi uzrok odstupanja, a utječe na 8 od 100 datoteka, odnosno na 16% datoteka sa različitim BN_i. Uslijed navedenih odstupanja, procedure SZZ i D'Ambros ponekad propuštaju očite veze, a ponekad uključuju veze s drugačijim inačicama ili iz drugih projekata.

Opažanje 10: Učinkovitost tehnika povezivanja nije konzistentna u različitom razvojnom okruženju. Ovo opažanje nadovezuje se na *opažanje 6*, ali sada je potvrđeno mjerom broja neispravnosti po datoteci. Uspoređujući skupove podataka dobivene primjenom procedura SZZ i alata BuCo, oni se podudaraju za više od 70% datoteka u inačicama projekta PDE, ali za manje od 45% datoteka u inačicama projekta JDT. Trend broja datoteka koje se podudaraju je također različit za ova dva projekta. U inačicama projekta JDT podudaranje kontinuirano opada od 72.3% na 43.9%, dok za PDE podudaranje prvo raste s 83.8% na 87.1%, da bi potom palo na 71.1%. Trend broja datoteka koje imaju različit broj neispravnosti je također različit. U inačicama projekta JDT broj datoteka u kategoriji *Manji* BN_i prvo raste sa 13.5% na 27.7%, da bi potom pao na 26.7%, dok za PDE uvijek iznosi 0%. U inačicama projekta JDT broj datoteka u kategoriji *Veći* BN_i konstantno raste od 14.2% do 29.5%, dok za projekt PDE inicijalno pada s

15.7% na 12.7%, da bi potom porastao na 28.9%. Za komponentu JDT Core nije moguće ustvrditi trend, budući da je procedura D'Ambros primijenjena za samo 1 inačicu. Ako razvojno okruženje promatramo na razini inačica unutar istog projekta, uočavamo da obje kategorije datoteka s nejednakim brojem neispravnosti imaju sličnu stopu rasta u 3 uzastopne inačice projekta JDT, a u inačicama projekta PDE ili nemaju značajnije promjene ili imaju nagle, skokovite promjene. Iz navedenoga se potvrđuje da razvojno okruženje utječe na učinkovitost povezivanja na razini projekata i na razini inačica.

5.3 Diskusija rezultata

Predstavljena komparativna studija provela je do sada najopsežniju usporedbu procedura prikupljanja podataka za predviđanje programskih neispravnosti i tehnika povezivanja razvojnih repozitorija. Budući da slična usporedba do sada nije provedena u istraživačkoj zajednici, predložene su mjere vrednovanja i procedure nužne za njezinu provedbu. Iz rezultata ove komparativne studije proizašlo je deset važnih opažanja glede učinkovitosti tehnika povezivanja i utjecaja razvojnog okruženja. Temeljem tih opažanja omogućeno je objašnjenje uzroka odstupanja promatranih procedura prikupljanja. Predstavljen je skup opažanja koje treba komentirati jer utječu na valjanost istraživanja i rezultata u studijama za SDP. Također, ta su nam opažanja omogućila dati odgovor na istraživačka pitanja postavljena u početku ovog poglavlja.

Rezultati komparativne studije ukazuju da **razvojno okruženje ima utjecaj na učinkovitost tehnika povezivanja**. Ovu spoznaju podupiru *opažanje 4*, *opažanje 5*, *opažanje 6* i *opažanje 10*. Utjecaj razvojnog okruženja može se promatrati kroz mjeru stope povezivanja, koju dvije najuspješnije tehnike povezivanja postižu unutar jedne razvojne zajednice i unutar inačica jednog njezinog projekta. Ustanovljeno je da trend stope povezivanja (*SP*) može značajno varirati unutar jedne razvojne zajednice. Primjerice, za projekte JDT i PDE mjera *SP* nikada nema padajući trend kojim ide ispod 70%, dok za projekt BIRT padajući trend može uzrokovati opadanje mjere *SP* s 80% na čak ispod 20%. Sva tri navedena projekta pripadaju istoj razvojnoj zajednici, Eclipse. Ustanovljeno je da stopa povezivanja može značajno varirati i unutar inačica jednog projekta. Primjerice, unutar projekta JDT varira u rasponu 49% - 93%, unutar projekta PDE u rasponu 22% - 95%, a unutar projekta BIRT u rasponu 18% - 80%.

Rezultati komparativne studije ukazuju da **tehnika povezivanja BuCo Regex pronalazi najveći broj točnih veza**. Uporaba procedura i tehnika koje sadrže prestrogo definirane krite-

rije ili koriste model predviđanja veza koje nedostaju ima negativan utjecaj na ishod povezivanja neispravnosti i programskog koda. Ovu spoznaju podupiru *opažanje 2*, *opažanje 3*, *opažanje 7*, *opažanje 8* i *opažanje 9*. Alat BuCo implementira proceduru prikupljanja koja nije stroga poput procedure SZZ, koja nužno zahtijeva da predaje sadrže određene ključne riječi unutar opisa, odnosno da neispravnosti budu prijavljene unutar 6 mjeseci od datuma izdanja neke inačice projekta. Ovaj uzrok odstupanja može uzrokovati promjenu u broju neispravnosti prisutnih u datotekama od 13% (za PDE 2.0) do čak 56% (za JDT 3.0). Tehnika BuCo Regex nije stroga poput tehnika vremenske korelacije i podudaranja autorstva, koje pronalaze mnogo manje neispravnosti. Ovaj uzrok odstupanja može uzrokovati promjenu u stopi povezivanja od 5% (za PDE 2.0) do čak 55% (za JDT 3.1). Tehnika BuCo Regex ne uključuje model predviđanja, kao što to čini alat ReLink. Ovaj uzrok odstupanja može uzrokovati promjenu u F-mjeri od 3% (za JDT 2.0) do 4.5% (za OpenNLP). Najvažnije, ovime su ostvareni izvorni znanstveni doprinosi definicije postupka za prikupljanje podataka s ciljem povećanja prikladnosti podataka i razvoja algoritma za prikupljanje podataka iz nestrukturiranih i formalno nepovezanih razvojnih repozitorija. Ovi doprinosi mogu poslužiti u procjeni valjanosti konstrukcije za buduća istraživanja i za procjenu rizika krive odluke uslijed nesigurnosti procedure prikupljanja podataka. Radi veće valjanosti budućih istraživanja, savjetuje se da se prikupljanje podataka provede alatom Buco, odnosno tehnikom BuCo Regex. Na taj će se način dobiti podaci visoke točnosti, odnosno postići će se veća valjanost konstrukcije. Postići će se i bolja usporedivost zaključaka, odnosno valjanost zaključaka istraživanja, jer će se temeljiti na istim osnovama.

Valjanost istraživanja

Ovo poglavlje komentira valjanost provedenog istraživanja sukladno preporukama iz poglavlja 1.3. Ponovljivost provedenog istraživanja, odnosno valjanost zaključaka, je značajna jer se temelji na javno dostupnim alatima i skupovima podataka te precizno definiranoj proceduri prikupljanja i usporedbe podataka. Ograničavajući faktor u poopćenju rezultata, odnosno vanjske valjanosti, predstavlja izvor podataka. Odabrani su projekti otvorenog koda, iz malog broja razvojnih zajednica, čija je domena primjene ograničena na razvojne, mrežne i poslovne programske sustave. Kako bi taj utjecaj bio što manji, odabrani su projekti različitih veličina iz dvije zajednice koji su ujedno bili predmet istraživanja i u mnogim srodnim studijama. Projekti razvojne zajednice Eclipse najčešće su korišteni upravo zbog veličine, dugotrajne evolucije te dostupnosti njezinih repozitorija. Odabir gotovih skupova podataka korištenih u komparativnoj

studiji ograničen je na one koji su javno dostupni, ali su zato korišteni svi koji su dostupni.

Otvoreni problem u istraživačkoj zajednici koji nije obuhvaćen ovom studijom jest problem veza koje nedostaju. Te veze prvenstveno se odnose na neispravnosti koje su zatvorene i zavedene kao ispravljene u sustavu za praćenje zahtjeva, a pri tome nisu povezane. Problematika je uočena u stopi povezivanja koja iznosi ispod 100%, kao što je prikazano tablicama 5.6, 5.7 i 5.8. Studije slučaja za SDP svakako bi trebale navoditi stopu povezivanja za podatke koje koriste jer to može biti indikator valjanosti konstrukcije njihova istraživanja. Veze koje nedostaju mogu se odnositi i na izostanak prijave neispravnosti unutar sustava za praćenje zahtjeva. Te veze nije moguće pronaći bez detaljnijeg uvida u aktivnosti programera, kao što je njihova međusobna korespondencija, ili bez njihove izravne pomoći. Tehnike temeljene na obradi prirodnog jezika i modelu predviđanja, kao što je alat ReLink, razvijene su s ciljem rješavanja navedenog problema. Međutim, ustanovljeno je da nepotpuno rješenje za ovaj problem može narušiti točnost podataka.

Poglavlje 6

Utjecaj razine neujednačenosti podataka

Brojne studije pokušale su pronaći najbolji model za SDP koristeći metode strojnog učenja. Uložen je velik trud u sistematizaciji postojećih saznanja, ali usporedivost rezultata dobivenih u različitim okruženjima i dalje je ograničena [39, 45, 113, 139, 168, 172]. Očekivano ponašanje većine metoda strojnog učenja je opadanje u performansama s porastom razine neujednačenosti skupova podataka. Posljednja analiza slučaja ove disertacije ima svrhu opisati utjecaj razine neujednačenosti skupova podataka na različite metode strojnog učenja za SDP. Provedeno je opsežno empirijsko istraživanje koje nastoji uvesti razinu neujednačenosti kao opis okruženja koji će poslužiti za dobivanje primjenjivih smjernica za SDP. Istraživanje se provodi nad skupovima podataka iz projekata otvorenog koda koji su dobiveni uporabom sustavno definirane procedure za prikupljanje podataka, predstavljenom u prethodnim poglavljima ove disertacije. Za potrebe empirijskog istraživanja definiran je i postupak za određivanje granične razine neujednačenosti skupova podataka. **Granična razina neujednačenosti** je pojam uveden ovom disertacijom, pod kojim se smatra onaj udio manjinske klase u skupu podataka od kojeg nastupa najveća relativna promjena odabrane mjere vrednovanja, s obzirom na vrijednosti te mjere u okolini. Manjinska klasa u skupovima podataka za SDP su moduli programskog sustava skloni neispravnostima (%SNP). Zato se ista oznaka (%SNP) koristi i za razinu neujednačenosti. Razina neujednačenosti je to veća što je %SNP manji. Empirijsko istraživanje postavlja sljedeća istraživačka pitanja:

1. Koliko iznosi granična razina neujednačenosti za pojedine metode strojnog učenja?
2. Koja je najuspješnija metoda strojnog učenja u različitim uvjetima razine neujednačenosti?

6.1 Metodologija empirijskog istraživanja

6.1.1 Materijal empirijskog istraživanja

Dizajn empirijskog istraživanja prati preporuke za provedbu analize slučaja [28]. Višestrukost izvora skupova podataka osigurana je uporabom dva dugovječna i velika projekta otvorenog koda, a višestrukost izvora rezultata osigurana je odabirom pet metoda strojnog učenja.

Odabrani skupovi podataka

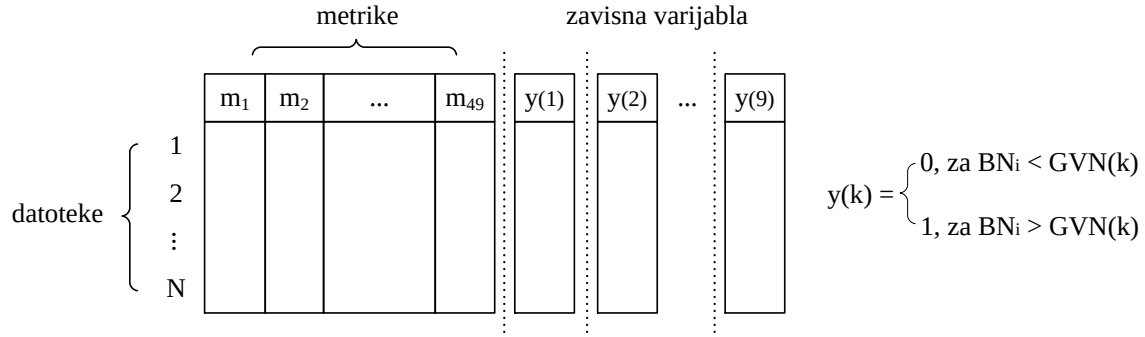
Podaci za ovo istraživanje prikupljeni su alatom BuCo, odnosno tehnikom povezivanja BuCo Regex, koja se pokazala najtočnijom u analizama iz poglavlja 5. Preuzeti su podaci iz višestrukih izvora, točnije iz 11 uzastopnih inačica projekata Eclipse JDT i Eclipse PDE. Svaki skup podataka predstavljen je jednom matricom. Programski moduli promatraju se na razini datoteka pa retci svake matrice predstavljaju datoteke unutar odabrane inačice projekta, a stupci predstavljaju vrijednosti metrika programskog koda. Broj redaka za svaki skup podataka prikazan je tablicom 6.1. Broj stupaca jednak je za sve skupove i sadrži 49 nezavisnih varijabli (metrika programskog koda) i 1 binarnu zavisnu varijablu (sklonost neispravnostima). Sadržane su sve metrike iz tablica P.1 i P.2 u prilogu P1, osim metrike *SUPER*, koja predstavlja ime nadrazreda te nema brojčanu vrijednost. Od metrika iz tablice P.3, u podešavanju parametara i testiranju modela temeljenih na metodama strojnog učenja koristi se samo *status*. Programski modul pripada klasi SNP ukoliko je broj njegovih neispravnosti veći od broja za graničnu vrijednost neispravnosti (GVN). Granična vrijednost neispravnosti mijenjana je u rasponu od 0 do 8. Manipulacijom broja te vrijednosti, u svakom skupu podataka mijenja se udio modula programskog sustava sklonih neispravnostima (%SNP), odnosno razina neujednačenosti. Sadržaj jednog skupa podataka i način na koji je dobiveno 9 skupova podataka iz svake inačice odabranih projekata prikazan je slikom 6.1.

Udio manjinske klase (%SNP) za svaki skup podataka prikazan je u tablici 6.1. Jedino za skupove podataka iz projekta PDE 3.8 postoji takva granična vrijednost neispravnosti da niti jedan modul ne spada u manjinsku klasu. S takvim skupom podataka nije moguće provesti podešavanje parametara modela za predviđanje te su, stoga, iz studije izbačeni svi skupovi koji pripadaju toj inačici.

Odabrane metode strojnog učenja

Za ovo empirijsko istraživanje odabrane su sljedeće metode strojnog učenja: logistička regresija, naivan Bayesov klasifikator, metoda potpornih vektora, slučajna šuma i rotirajuća šuma. Motivacija za korištenje ovih metoda predstavljena je u poglavlju 2.2.2. Riječ je o provjerenim i pouzdanim metodama ili novijim metodama koje su postigle odlične rezultate u različitim domenama primjene. U uvjetima neujednačenih skupova podataka njihovi rezultati mogu biti različiti. Neke metode postižu dobre rezultate, za neke metode poznato je da daju loše rezultate kod jako visokih razina neujednačenosti, a neke metode nisu još analizirane u tom okruženju. Međutim, niti za jednu nije poznata točna granica kod koje nastupa najveća promjena njezinih performansi, odnosno nije poznata granična razina neujednačenosti.

Parametri modela odabranih metoda strojnog učenja su podešeni i vrednovani u *eksperimentalnom okružju* (eng. experimental environment) programskog alata Weka, inačica 3.6.9 [173]. Weka je popularni programski alat otvorenog koda za primjenu strojnog učenja. Razvijen je u programskom jeziku Java, na sveučilištu Waikato s Novog Zelanda. Metode logističke regresije i naivnog Bayesovog klasifikatora nisu zahtijevale odabir dodatnih parametara. Metoda potpornih vektora korištena je sa radijalnom jezgrenom funkcijom uz parametar Gamma ($\gamma = 10$), sukladno preporuci srodnog istraživanja [102]. Metoda slučajne šume koristi preporučenu vrijednost od 100 stabala, a svako stablo dobije $\sqrt{n}$ metrika, pri čemu n predstavlja ukupni broj metrika u skupu podataka [124]. Rotirajuća šuma je najnovija metoda, za koju još nema preporuka za konfiguraciju njezinih parametara. Budući da je već bila istraživana u radu [38], korištene su zadane postavke, za koje je ustanovljeno da nisu značajno nadmašene drugim kombinacijama njihovih vrijednosti. Grupe metrika su maksimalno veličine $M = 3$, a podešavanje parametara provodi se u 10 iteracija.



Slika 6.1: Sadržaj matrica skupova podataka korištenih u empirijskom istraživanju

Tablica 6.1: Broj datoteka i udio manjinske klase %SNP u skupovima podataka

Inačica	Broj	Udio manjinske klase %SNP za odabranu GVN								
Projekta	Datoteka	0	1	2	3	4	5	6	7	8
JDT 2.0	2397	46%	26%	18%	12%	9%	7%	5%	4%	3%
JDT 2.1	2743	32%	16%	9%	6%	4%	2.7%	2.1%	1.5%	1.2%
JDT 3.0	3420	39%	23%	14%	10%	7%	6%	4%	3%	2.7%
JDT 3.1	3883	33%	18%	12%	8%	6%	5%	4%	2.9%	2.3%
JDT 3.2	2233	37%	20%	12%	8%	6%	4%	3%	2.9%	1.8%
JDT 3.3	4821	24%	11%	6%	4%	2.2%	1.3%	1.0%	0.9%	0.7%
JDT 3.4	4932	19%	6%	3%	1.9%	1.1%	0.7%	0.5%	0.5%	0.4%
JDT 3.5	4395	11%	4%	2.3%	1.3%	0.9%	0.6%	0.3%	0.2%	0.2%
JDT 3.6	4392	9%	3%	1.2%	0.6%	0.4%	0.2%	0.1%	0.1%	0.1%
JDt 3.7	4415	9%	3%	1.5%	1.0%	0.5%	0.3%	0.2%	0.2%	0.2%
JDT 3.8	4444	7%	2.8%	1.7%	1.1%	0.9%	0.7%	0.5%	0.3%	0.3%
PDE 2.0	576	19%	8%	4%	2.4%	1.9%	1.6%	1.2%	1.2%	0.7%
PDE 2.1	761	16%	5%	4%	2.1%	1.1%	0.5%	0.5%	0.3%	0.3%
PDE 3.0	881	31%	13%	7%	4%	2.5%	1.2%	1.2%	0.8%	0.8%
PDE 3.1	1108	32%	15%	9%	4%	2.8%	1.8%	1.0%	0.7%	0.5%
PDE 3.2	1351	46%	17%	7%	4%	2.3%	1.2%	0.7%	0.7%	0.6%
PDE 3.3	1713	44%	23%	13%	8%	6%	4%	2.9%	2.2%	1.7%
PDE 3.4	2144	28%	15%	9%	6%	4%	4%	2.9%	2.5%	2.1%
PDE 3.5	2297	32%	17%	11%	8%	6%	4%	3%	2.5%	1.7%
PDE 3.6	2412	16%	6%	2.8%	1.8%	1.3%	0.9%	0.6%	0.5%	0.4%
PDE 3.7	2404	27%	4%	1.5%	0.7%	0.3%	0.2%	0.2%	0.1%	0.0%
PDE 3.8	3522	2%	0.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Vrednovanje modela predviđanja

Vrednovanje modela predviđanja temelji se na matrici zabune, opisanoj u poglavlju 2.2.3. Budući da se analizira ponašanje metoda strojnog učenja u okruženju neujednačenih skupova podataka, kao najprikladnija odabrana je mjera geometrijske sredine (GM) [130]. Za razliku od mnogo drugih mjera, geometrijska sredina (GM) obuhvaća i pogrešku tipa 1 i pogrešku tipa

2. Na taj način objedinjuje oba sukobljena cilja u jednoj mjeri, što je važno za neujednačene skupove podataka [128]. Mjera Ave također objedinjuje oba sukobljena cilja, ali geometrijska sredina (GM) brže propagira jer koristi njihovu geometrijsku sredinu i zato je osjetljivija na razlike u točnosti manjinske i većinske klase. Rezultati mjere geometrijske sredine (GM) su grupirani s obzirom na udio manjinske klase (%SNP) u skupu podataka.

6.1.2 Metoda za određivanje utjecaja razine neujednačenosti

U srodnoj literaturi nije pronađen primjer istraživanja koji analizira ponašanje modela predviđanja s obzirom na promjenjivu razinu neujednačenosti skupova podataka. Pojedine studije navele su da problem neujednačenosti postoji pokušale ga ublažiti uporabom određenih tehnika za predobradu podataka, ali to se nije pokazalo kao univerzalno rješenje [105]. Ostalo je neistraženo u kojim se uvjetima taj problem značajnije manifestira te postoji li okvir uvjeta unutar kojih se mogu očekivati konzistentne performanse. Ova disertacija predlaže metodu za određivanje utjecaja razine neujednačenosti koja započinje sljedećim koracima:

(1) Postaviti graničnu vrijednost neispravnosti (GVN) na uobičajenu vrijednost 0 i izračunati udio manjinske klase (%SNP) za dobiveni skup podataka;

(2) Uporabom *unakrsne provjere u 3 preklopa* (eng. 3-fold cross validation) stratificirano uzorkovati podatke na skup za učenje i skup za testiranje;

(3) Izgraditi model predviđanja nad skupom za učenje i vrednovati nad skupom za testiranje;

(4) Vrednovanje provesti mjerom geometrijske sredine (GM);

(5) Ponoviti korake (1) - (4) 20 puta;

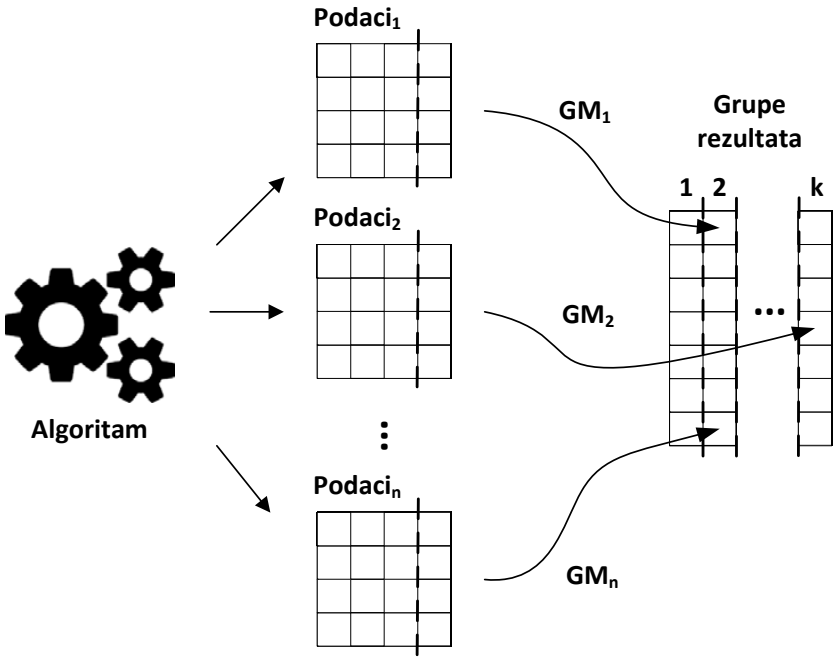
(6) Uvećati graničnu vrijednost neispravnosti (GVN) za 1 i ponoviti korake (1) - (5), dok udio manjinske klase (%SNP) ne dosegne vrijednosti ispod 1%;

(7) Grupirati rezultate izražene u mjeri GM, s obzirom na udio manjinske klase (%SNP);

(8) Testirati jesu li grupe rezultata mjere GM normalno distribuirane;

- (a) Testirati imaju li normalno distribuirane grupe rezultata homogene varijance;
- (b) Testirati jesu li normalno distribuirane grupe rezultata ujedno i sferične.

Nakon provedenih koraka (1) - (8), može se provesti analiza s ciljem utvrđivanja granične razine neujednačenosti skupova podataka i analiza s ciljem određivanja metode strojnog učenja čije su performanse najbolje za određenu razinu neujednačenosti skupova podataka.



Slika 6.2: Grupiranje rezultata za utvrđivanje granične razine neujednačenosti

Utvrđivanje granične razine neujednačenosti

Utvrđivanje granične razine neujednačenosti odnosi se na svaku metodu strojnog učenja odvojeno. Rezultati, izraženi u mjeri geometrijske sredine (GM) i dobiveni unakrsnom provjerom, grupiraju se u k grupa rezultata, kako je prikazano slikom 6.2. Svaka od k grupa predstavlja određeni raspon udjela manjinske klase (%SNP) koji je bio prisutan u skupu podataka za koji je dobiven određeni rezultat. Primjerice, ako za skup podataka i vrijedi $\%SNP_i < 3\%$, tada rezultat GM_i ulazi u grupu rezultata $k = 1$ za koju vrijedi $\%SNP < 3\%$. Nasumičnim i stratificiranim uzorkovanjem na skup za učenje i skup za testiranje očekuju se različiti skupovi podataka. To znači da se svaki skup podataka koristi samo jednom, odnosno da grupe rezultata nisu zavisne. Pored toga, grupe podataka ne moraju imati jednak broj članova. Predložena metoda za utvrđivanje granične razine neujednačenosti nastavlja se u sljedećim koracima:

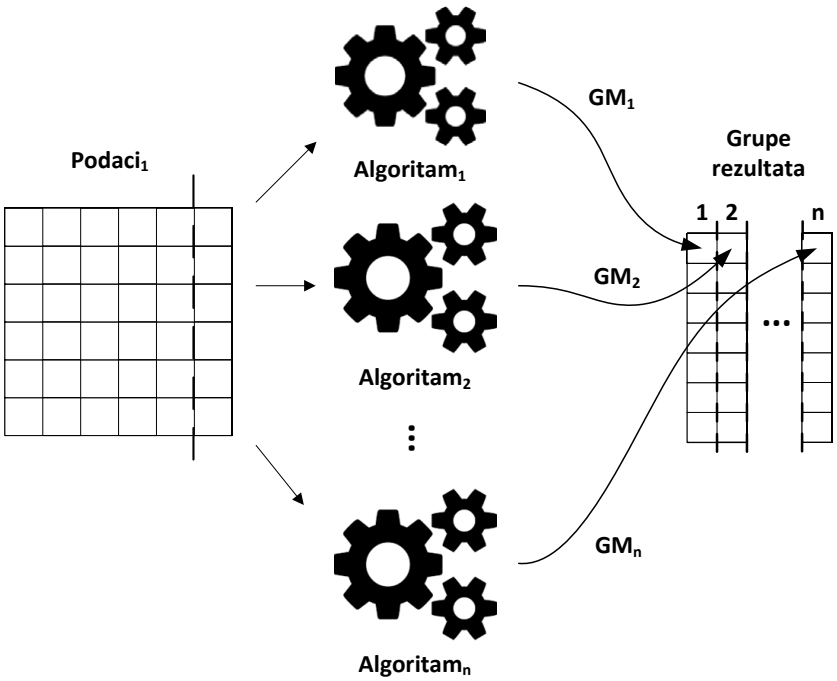
(9) Testirati postoje li značajne razlike među grupama mjere GM koristeći:

(a) *Jednosmjernu analizu varijance* (eng. one-way ANOVA) za grupe rezultata normalne distribucije čije varijance su homogene;

(b) Welchovu analizu varijance (ANOVA) za grupe rezultata normalne distribucije čije varijance nisu homogene;

(c) Neparametarski test Kruskal-Wallis za rezultate koji nisu normalne distribucije.

(10) Ukoliko značajna razlika postoji, odrediti ovisnost rezultata pojedine metode strojnog



Slika 6.3: Grupiranje rezultata za usporedbu metoda strojnog učenja

učenja izraženih u mjeri GM i udjela manjinske klase (%SNP) koristeći:

(a) Srednje vrijednosti grupa mjere GM za normalno distribuirane rezultate;

(b) Medijane grupa mjere GM za rezultate koji nisu normalne distribucije.

(11) Aproksimirati funkciju $GM = f(\%SNP)$ polinomom višeg reda;

(12) Izračunati prvu derivaciju $f'(\%SNP)$ i drugu derivaciju $f''(\%SNP)$;

(13) Izračunati Arrow - Prattovu mjeru izrazom [174, 175]:

$$A(\%SNP) = -\frac{f''(\%SNP)}{f'(\%SNP)} \tag{6.1}$$

(14) Pronaći udio manjinske klase (%SNP) za koji A(%SNP) ima najveću vrijednost u predjelu u kojem je f(%SNP) monotono rastuća;

(15) Dobivena vrijednost predstavlja **graničnu razinu neujednačenosti**, a postupak ponoviti za odabrane metode strojnog učenja.

Usporedba metoda strojnog učenja s obzirom na razine neujednačenosti

Usporedba metoda strojnog učenja s obzirom na razine neujednačenosti odnosi se na primjenu različitih metoda strojnog učenja nad istim podacima. Rezultati, izraženi u mjeri GM i dobiveni unakrsnom provjerom, grupiraju se u n grupa rezultata, kako je prikazano slikom 6.3. Svaka od n grupa predstavlja rezultate dobivene jednom od n metoda strojnog učenja. Istovremeno

se pamti i udio manjinske klase (%SNP) koji je bio prisutan u korištenim skupovima podataka, kako bi se metode strojnog učenja mogle usporediti u različitim uvjetima. Budući da se metode strojnog učenja koriste nad istim podacima, grupe rezultata nisu nezavisne, ali sadrže jednak broj članova. Predložena metoda za usporedbu metoda strojnog učenja s obzirom na razine neujednačenosti u skupovima podataka nastavlja se u sljedećim koracima:

(9) Provesti korake (1) - (8) za odabrane metode strojnog učenja;

(10) Testirati postoji li značajna razlika među odabranim metodama strojnog učenja za svaku grupu rezultata, koja je grupirana s obzirom na udio manjinske klase (%SNP), koristeći:
   (a) Jednosmjernu analizu varijance (ANOVA) s *ponovljenim mjerenjima* (eng. repeated measures) za grupe rezultata normalne distribucije koje su i sferične;
   (b) Jednosmjernu analizu varijance (ANOVA) s ponovljenim mjerenjima i Greenhouse-Geisser korekcijom za grupe rezultata normalne distribucije koji nisu sferične;
   (c) Neparametarski test Friedman za rezultate koji nisu normalne distribucije.

(11) Ukoliko značajna razlika postoji, odrediti ovisnost rezultata pojedine metode strojnog učenja GM i udjela manjinske klase (%SNP) koristeći:
   (a) Srednje vrijednosti grupa mjere GM za normalno distribuirane rezultate;
   (b) Medijane grupa mjere GM za rezultate koji nisu normalne distribucije.

(12) Provesti *post hoc* analizu radi određivanja ranga metoda strojnog učenja za svaki udio manjinske klase (%SNP).

Unakrsna provjera

Prvih nekoliko koraka u predloženoj metodi predstavlja *n-puta ponovljenu unakrsnu provjeru u k-preklopa* (eng. *n*-times *k*-fold cross validation) [95]. Tim se postupkom skup podataka uzorkuje u k preklopa jednake veličine. Primjenom stratificiranog pristupa uzorkovanju podataka, osigurava se da svi preklopi imaju jednaku zastupljenost obje klase podataka. To je vrlo važno u slučaju visokih razina neujednačenosti jer bi se u protivnome lako moglo dogoditi da neki preklop nema niti jednog pripadnika manjinske klase. Nakon uzorkovanja, parametri modela predviđanja se podešavaju nad $k-1$ preklopa, a testira se nad jednim preostalim preklopom. Postupak se ponavlja k puta, tako da se testiranje svaki put provede nad idućim preklopom. Cijela procedura ponavlja se n puta, tako da se nasumičnom podjelom svaki put dobije preklope drugačijeg sastava. Predložena je podjela u 3 preklopa, tako da preklopi budu dovoljno veliki (33% ukupnog skupa podataka) kako bi se smanjila vjerojatnost pojave preklopa koji nema niti

jednog pripadnika manjinske klase. Postupak se ponavlja 20 puta kako bi se dobilo dovoljno velik uzorak za provedbu statističkih testova i kako bi se osiguralo dobivanje pouzdanijih zaključaka.

Statistički testovi

Rezultati su predstavljeni mjerom GM jer se ona smatra najboljom mjerom za analizu performansi modela predviđanja neujednačenih skupova podataka [129, 130]. Nakon grupiranja rezultata s obzirom na udio manjinske klase (%SNP) potrebno je provjeriti prate li rezultati normalnu distribuciju. Ako prate, potrebno je provjeriti jesu li varijance jednake (homogene) između grupa rezultata koji se uspoređuju, odnosno je li zadovoljen uvjet sferičnosti rezultata. To su važni podaci koji utječu na odabir odgovarajućeg statističkog *općeg* (eng. omnibus) testa, koji će ustanoviti postoji li barem jedan par grupa rezultata koji je značajno različit [128]. Za nezavisne grupe rezultata na raspolaganju su:

- Jednosmjerna ANOVA, koja polazi od pretpostavke da su grupe rezultata normalne distribucije i homogenih varijanci [128];
- Welch ANOVA, koja polazi od pretpostavke da su grupe rezultata normalne distribucije, ali da im varijance nisu homogene [176];
- Kruskal-Wallis, neparametarski test koji polazi od pretpostavke da su grupe rezultata međusobno nezavisne [128].

Za zavisne grupe rezultata na raspolaganju su:

- Jednosmjerna ANOVA s ponovljenim mjerenjima, koja polazi od pretpostavke da su grupe rezultata normalne distribucije i sferične [128];
- Jednosmjerna analiza varijance (ANOVA) s ponovljenim mjerenjima i Greenhouse-Geisser korekcijom, koja polazi od pretpostavke da su grupe rezultata normalne distribucije i da nisu sferične [176];
- Friedmanov test, neparametarski test koji polazi od pretpostavke da su grupe rezultata međusobno zavisne [128].

Shapiro-Wilkov test jedan je od mogućih testova koji provjerava prati li neka grupa mjere GM normalnu distribuciju. Usporedbom popularnih testova normalnosti, Shapiro-Wilkovog, Kolmogorov–Smirnovljevog, Lillieforsovog, i Anderson–Darlingovog, ustanovljeno je da Shapiro-Wilkov ima najveću statističku snagu [177]. Testom se dobije p-vrijednost kojom se potvrdi ili opovrgne nul-hipoteza da su podaci normalno distribuirani uz odabranu razinu značajnosti

$\alpha = 0.05$. Leveneov test homogenosti varijanci jedan je od mogućih testova čija je nul-hipoteza da su varijance populacije jednake. Mauchlyev test sferičnosti jedan je od mogućih testova čija je nul-hipoteza da su razlike među varijancama jednake za sve parove grupa.

Jednosmjerna ANOVA je parametarski test kojim se provjerava nul-hipoteza da grupe rezultata mjere GM potječu iz iste populacije na temelju njihovih srednjih vrijednosti. ANOVA predstavlja ekstenziju Studentovog t-testa za više od dvije grupe nezavisnih rezultata. Ukoliko grupe rezultata nisu nezavisne, koristi se jednosmjerna ANOVA s ponovljenim mjerenjima, koja razlikuje varijaciju rezultata za isti skup i varijaciju rezultata između različitih skupova [128]. Uporabom statističkog F-testa dobije se p-vrijednost kojom se potvrdi ili opovrgne nul-hipoteza [178].

Kruskal-Wallis je neparametarski test i inačica jednosmjerne ANOVA-e [128], a predstavlja ekstenziju Mann-Whitneyevog testa za više od dvije grupe nezavisnih rezultata [176]. Ovim testom se provjerava nul-hipoteza da grupe rezultata mjere GM potječu iz iste populacije na temelju njihovih medijana. Mann-Whitneyev test pritom sortira podatke uzlazno i koristi njihov rang umjesto korištenja njihovih numeričkih vrijednosti [176]. Aproksimacijom s hi-kvadrat (χ^2) distribucijom dobije se p-vrijednost kojom se potvrdi ili opovrgne nul-hipoteza [179].

Friedman je neparametarski test i inačica jednosmjerne ANOVA-e s ponovljenim mjerenjima [128], a predstavlja ekstenziju Wilcoxonovog *signed-rank* testa [176]. Ovim testom provjerava se nul-hipoteza da grupe rezultata mjere GM potječu iz iste populacije na temelju njihovih medijana. Preciznije rečeno, nul-hipoteza je da je distribucija razlika između rezultata simetrična oko vrijednosti nula [176]. Wilcoxonov *signed-rank* test pritom sortira razlike zavisnih mjerenja uzlazno, rangira ih neovisno o predznaku i potom svakom rangu dodjeljuje predznak koji odgovara razlici mjerenja. Aproksimacijom s hi-kvadrat (χ^2) distribucijom, dobije se p-vrijednost kojom se potvrdi ili opovrgne nul-hipoteza [176].

Za slučaj da se opovrgne nul-hipoteza ANOVA-e, Kruskal-Wallisovog ili Friedmanovog testa, može se provesti i *post hoc* analiza kako bi se ustanovilo koje se grupe podataka mjere GM značajno razlikuju od kojih [128]. Za potrebe usporedbe metoda strojnog učenja, s obzirom na razine neujednačenosti, *post hoc* analiza je neophodna. Budući da grupe rezultata nisu jednake veličine, nakon ANOVA-e se koristi Scheffeov *post hoc* test. *Post hoc* analizom se testiraju svi parovi grupa mjere GM, zbog čega treba uporabiti i Holm-Bonferronijevu korekciju p-vrijednosti [128, 180]. Nakon provedene *post hoc* analize, moguće je odrediti rang svake od metode strojnog učenja tako da se poredaju od najboljeg (1) do najgoreg (5). Rang se određuje

s obzirom na broj metoda koje imaju značajno lošije rezultate te se razlikuje nekoliko slučajeva:

- Ukoliko su sve metode značajno različite, rang je jednak poretku određene metode. Primjerice, rang može iznositi: **1, 2, 3**;
- Ukoliko dvije metode nemaju značajno različite rezultate, one dijele isti rang. Metoda koja je značajno lošija poprima rang koji odgovara njezinom poretku. Primjerice, rang može iznositi: **1, 2, 2, 4**;
- Ukoliko dvije metode imaju značajno različite rezultate, a postoji treća metoda koja je rezultatom između njih i nije značajno različita, tada treća metoda dijeli rang s boljom metodom, a lošija metoda ima rang koji je veći od njezinog poretka. Primjerice, rang može iznositi: **1, 2, 2, 3**.

Utvrđivanje granične razine neujednačenosti

Ukoliko statistički test ukaže na značajnu razliku među rezultatima modela predviđanja u ovisnosti o razini neujednačenosti skupova podataka, može se odrediti gdje nastupaju najveće razlike. Potrebno je odrediti ovisnost medijana ili srednje vrijednosti mjere GM u odnosu na udio manjinske klase u skupu podataka (%SNP). Aproksimaciju funkcije $GM = f(\%SNP)$ može se dobiti uporabom polinoma višeg reda. Radi postizanja čim bolje aproksimacije i budući da se za izračun granične razine koriste prva i druga derivacija te funkcije, predlaže se da polinom bude barem petog reda. Izračunom prve derivacije funkcije $f'(\%SNP)$, dobije se krivulja nagiba aproksimirane krivulje. Krivulja nagiba ukazuje na brzinu promjene performansi modela za predviđanje, ali i dalje ne otkriva kritičnu točku u kojoj nastupa najveća promjena. Izračunom druge derivacije funkcije $f''(\%SNP)$ dobije se brzina promjene aproksimirane krivulje. Da bi se pronašlo "točku koljena", odnosno točku nagle promjene, potrebna nam je mjera relativne promjene mjere GM s obzirom na okolne vrijednosti. Način na koji se to može učiniti je izračunom Arrow-Prattove mjere A(%SNP), koristeći jednadžbu 6.1 [174, 175]. Mjera A(%SNP) ima to veći iznos što je veća zakrivljenost aproksimirane krivulje u nekoj točki razine neujednačenosti. Ova mjera je originalno definirana kao mjera *apsolutne sklonosti izbjegavanja rizika* (eng. Absolute Risk Aversion), s obzirom na funkciju korisnosti uloženog novca [174, 175]. Kako je njezina uporaba ograničena na monotono rastuće funkcije, potrebno ju je ograničiti na područje unutar kojeg vrijedi: $f'(\%SNP) > 0$. Granična razina neujednačenosti utvrđuje se pronalaskom udjela %SNP za koji funkcija A(%SNP) ima najveću vrijednost.

Granična razina neujednačenosti može se utvrditi i za model predviđanja koji ima bo-

lje ponašanje u uvjetima visoke razine neujednačenosti. Tada izraz za Arrow-Prattove mjeru A(%SNP) iz jednadžbe 6.1 nema negativan predznak, a pretragu ograničimo na područje za koje vrijedi: $f'(\%SNP) < 0$. Interpretacija tako pronađene granične razine neujednačenosti se mijenja jer ona ukazuje na razinu neujednačenosti kod koje performanse određene metode strojnog učenja počinju naglo rasti zajedno s porastom razine neujednačenosti.

6.2 Rezultati empirijskog istraživanja

Opisi rezultata provedenog empirijskog istraživanja podijeljeni su u dva dijela, s obzirom na dva istraživačka pitanja postavljena u uvodu poglavlja 6. Podjela rezultata mjere geometrijske sredine (GM) u grupe s obzirom na razinu neujednačenosti te provedba testova kojim se provjerava jesu li te grupe normalno distribuirane zajedničke su za oba dijela. Obrada rezultata obavljena je u programskom paketu Matlab, inačica R2014a [181] i Statistica, inačica 8 [182].

Prateći korake predložene metode za analizu utjecaja različitih razina neujednačenosti podataka, prvo je provedeno podešavanje i vrednovanje parametara modela odabranih metoda strojnog učenja 20 puta ponovljenom unakrsnom provjerom u 3 preklopa. Dobiveno je 60 vrijednosti mjere GM za svaki skup podataka i za svaku metodu strojnog učenja. Jedan skup podataka predstavlja jednu inačicu projekta za odabranu graničnu vrijednost neispravnosti (GVN). Rezultati izraženi u mjeri GM potom su grupirani s obzirom na udio modula programskog sustava %SNP koji se nalazi u skupu podataka iz kojeg su rezultati dobiveni. Grupiranje je prvo provedeno za 11 inačica projekta JDT (2.0 - 3.8), potom za 10 inačica projekta PDE (2.0 - 3.7), a u konačnici i za svih 21 skupova zajedno. Broj skupova podataka i broj uzoraka koji je ušao u pojedinu grupu rezultata prikazani su tablicom 6.2. Tablica 6.2 je kompaktniji način zapisa iz tablice 6.1. Na primjer, jedini skup podataka iz projekta JDT koji se nalazi unutar raspona 40% - 50% u tablici 6.2 je inačica JDT 2.0 pri vrijednosti GVN = 0 iz tablice 6.1. Iz svega navedenog slijedi da je broj uzoraka u svakoj grupi jednak umnošku broja skupova podataka iz tablice 6.2 i broja 60, koliko se vrijednosti mjere GM dobije 20 puta ponovljenom unakrsnom provjerom u 3 preklopa. Na primjer, veličina grupe rezultata (za jednu metodu strojnog učenja) unutar raspona 40% - 50% za projekt JDT iznosi 60, a za projekt PDE iznosi 120.

Nakon grupiranja rezultata proveden je test normalnosti Shapiro-Wilk, uz razinu značajnosti $\alpha = 0.05$. Nul-hipoteza ovog testa je da grupe rezultata mjere GM, koje su grupirane s obzirom na razinu neujednačenosti u skupovima podataka iz kojih su proizašle, prate normalnu distribuciju. Testovi su pokazali da niti jedna grupa rezultata izraženih mjerom GM ne prati normalnu distribuciju. Za svaki provedeni test, p-vrijednost je bila značajno manja od 0.01. Posljedično, sve naredne analize provode se nad medijanima i koriste se neparametarski statistički testovi za provjeru značajnosti. Radi boljeg uvida u distribuciju podataka i njezino odstupanje od normalne razdiobe, provedena je dodatna analiza. Tablica P.4 u prilogu P2 prikazuje koliko iznose srednja vrijednost, standardna devijacija, medijan, interkvartilni raspon, zakrivljenost i spljoštenost svake grupe rezultata izraženih mjerom GM.

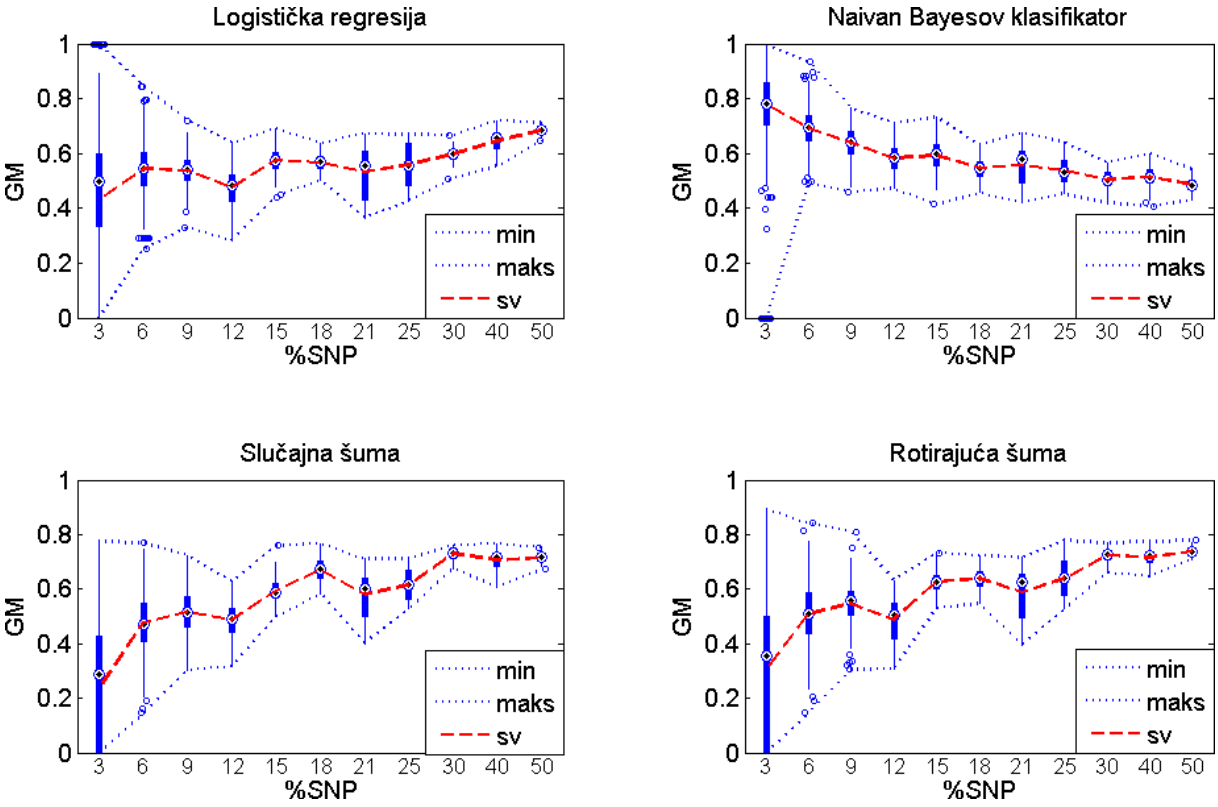
Tablica 6.2: Broj skupova podataka i (broj uzoraka) u svakoj grupi rezultata mjere GM koje su podijeljene s obzirom na udio manjinske klase (%SNP)

Raspon %SNP	**JDT**	**PDE**	Ukupno
0% - 3%	49 (2940)	49 (2941)	98 (5880)
3% - 6%	20 (1200)	15 (900)	35 (2100)
6% - 9%	8 (480)	8 (480)	16 (960)
9% - 12%	6 (360)	1 (60)	7 (420)
12% - 15%	3 (180)	4 (240)	7 (420)
15% - 18%	2 (120)	4 (240)	6 (360)
18% - 21%	3 (180)	1 (60)	4 (240)
21% - 25%	2 (120)	1 (60)	3 (180)
20% - 30%	1 (60)	2 (120)	3 (180)
30% - 40%	4 (240)	3 (180)	7 (420)
40% - 50%	1 (60)	2 (120)	3 (180)
Zbroj:	99 (5940)	90 (5400)	189 (11340)

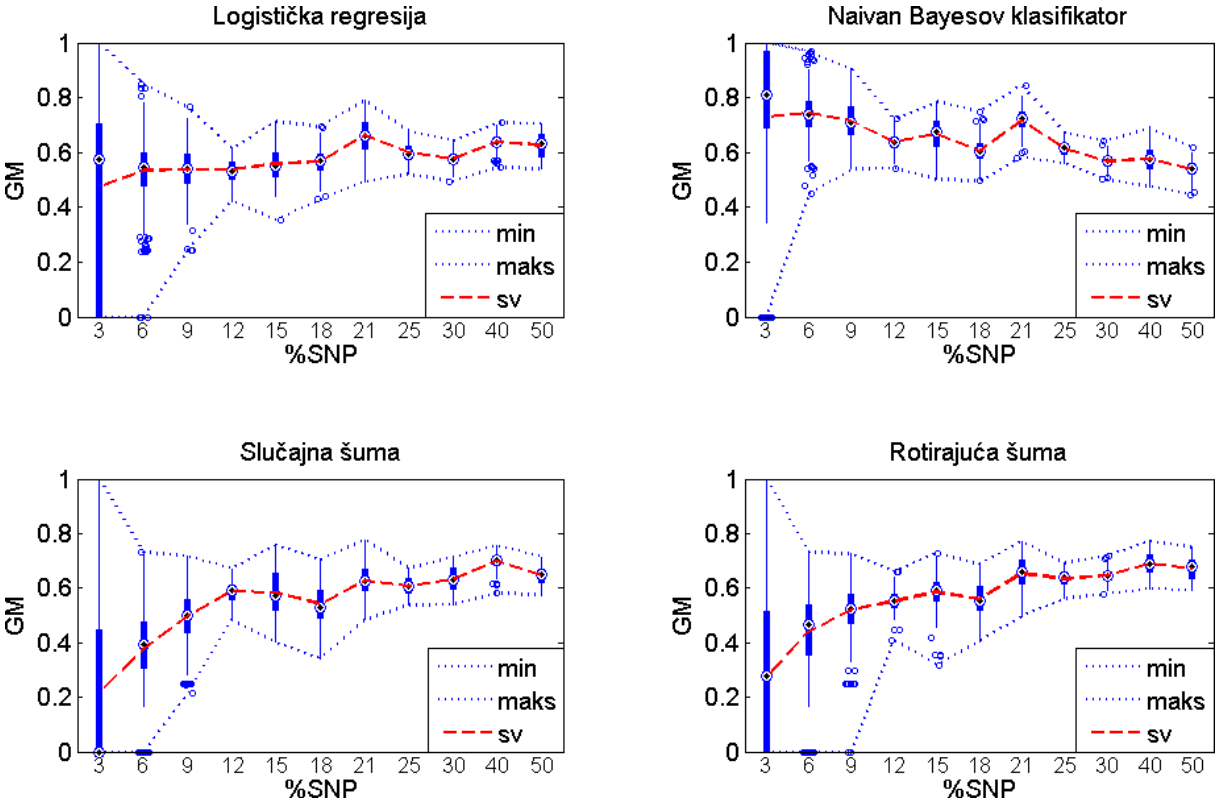
6.2.1 Granična razina neujednačenosti

Prva analiza s ciljem utvrđivanja granične razine neujednačenosti podataka podrazumijeva deskriptivnu statistiku dobivenih grupa rezultata izraženih mjerom GM. Korišten je *pravokutni dijagram* (eng. box and whisker diagram) koji grafički predstavlja mjeru centralne tendencije i mjeru rasipanja podataka. Centar pravokutnika predstavlja medijan, rubovi pravokutnika predstavljaju interkvartilni raspon, vrhovi linija predstavljaju puni raspon, a kružićima su predstavljene stršeće vrijednosti. Slike 6.4 i 6.5 prikazuju pravokutne dijagrame za svaku grupu rezultata GM za skupove podataka iz projekata JDT i PDE, a slika 6.6 za oba projekta zajedno. Pored pravokutnih dijagrama, crvenom isprekidanom linijom je označena i srednja vrijednost (sv), a plavim točkastim linijama su potvrđene minimalne (min) i maksimalne (maks) vrijednosti.

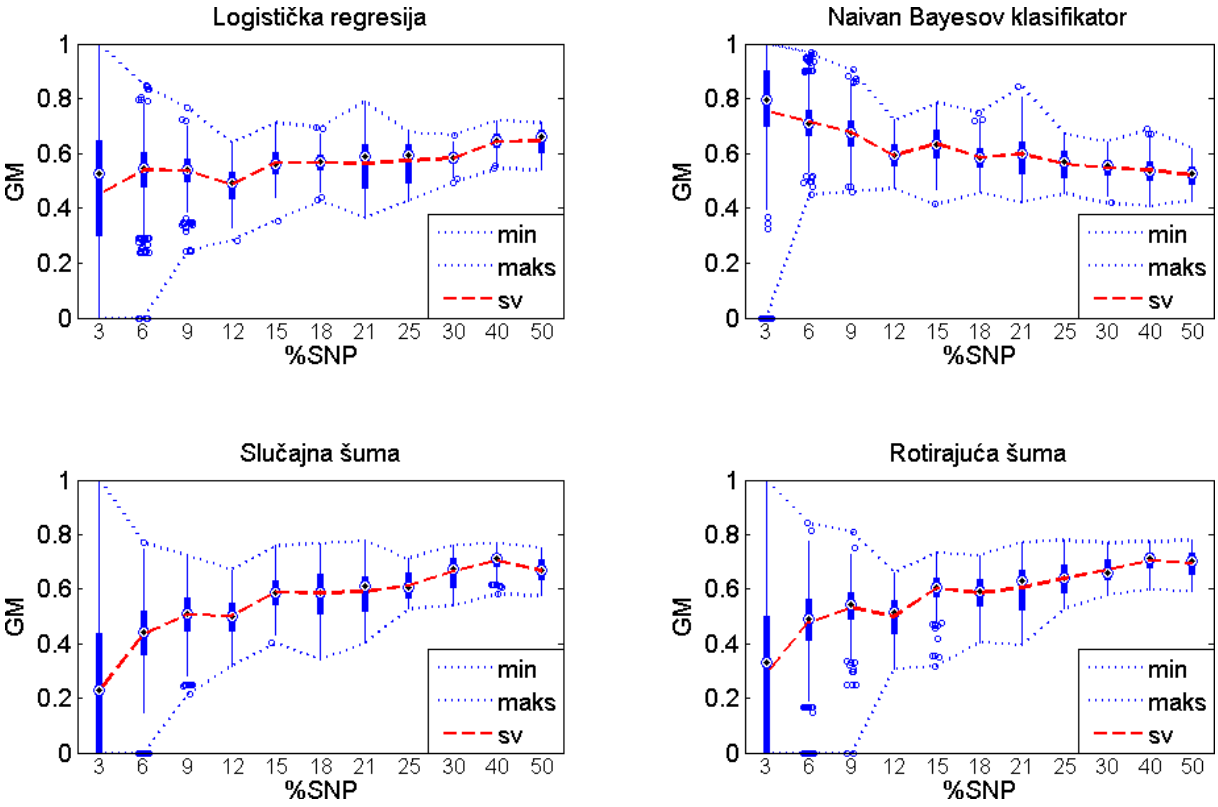
Najveća razina neujednačenosti skupova podataka je u ishodištu x-osi, kada se udio manjinske klase (%SNP) približava vrijednosti 0. Uvidom u rezultate prikazane slikama 6.4, 6.5 i 6.6 uočava se postepena degradacija performansi svih metoda strojnog učenja s povećanom razinom neujednačenosti. Za logističku regresiju, slučajnu šumu i rotirajuću šumu, vrijednosti mjere GM sve više opadaju s porastom neujednačenosti. Iznenađujuće, za metodu naivnog Bayesovog klasifikatora vrijednosti mjere GM se povećavaju s porastom neujednačenosti. Za-



Slika 6.4: Deskriptivna analiza grupa rezultata mjere GM u skupovima projekta JDT



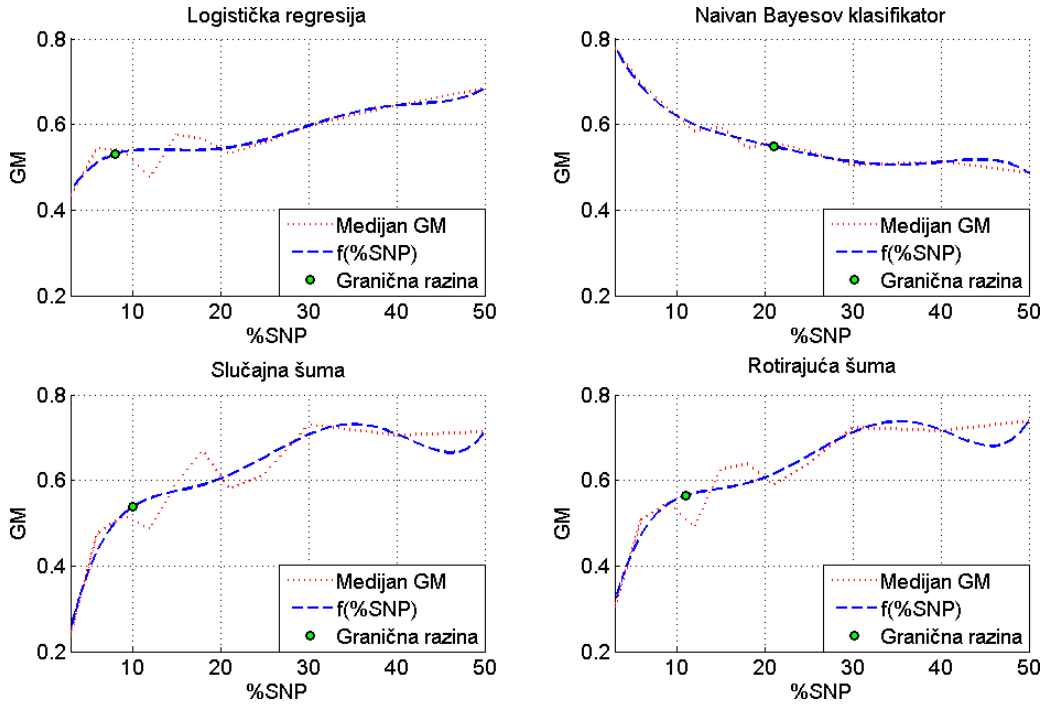
Slika 6.5: Deskriptivna analiza grupa rezultata mjere GM u skupovima projekta PDE



Slika 6.6: Deskriptivna analiza grupa rezultata mjere GM u skupovima projekta JDT i PDE

jedničko svim metodama je da s porastom neujednačenosti njihove performanse postaju nestabilnije. Interkvartilni raspon počinje biti sve veći, a ukupni raspon pokriva gotovo cijeli mogući raspon od 0 do 1. Interesantno je uočiti i da je razlika između srednje vrijednosti (sv) i medijana (centar pravokutnika) vrlo mala.

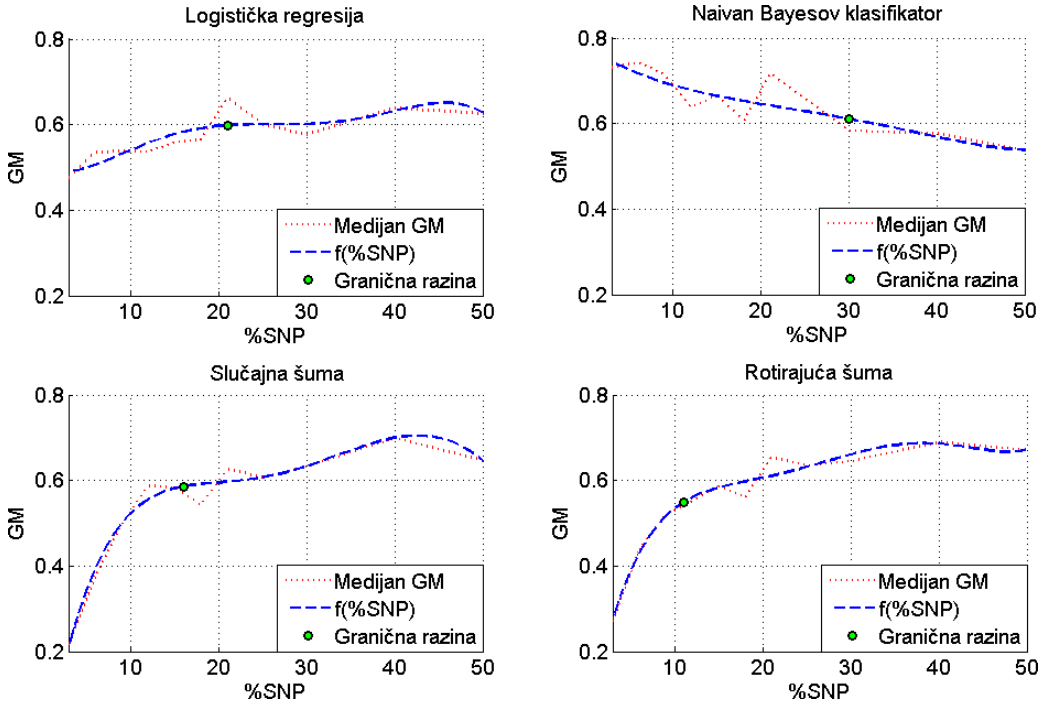
Idući korak je statističko testiranje kojim će se odrediti postoji li značajna razlika između grupa mjere GM, grupiranih s obzirom na udio manjinske klase (%SNP). Budući da rezultati nisu normalno distribuirani, korišten je Kruskal-Wallisov neparametarski test, uz razinu značajnosti $\alpha = 0.05$. Nul-hipoteza ovog testa je da ne postoji značajna razlika između grupa rezultata izraženih mjerom GM, koje su grupirane s obzirom na razinu neujednačenosti u skupovima podataka iz kojih su proizašle. Drugim riječima, nul-hipoteza je da, za određenu metodu strojnog učenja, ne postoji značajna razlika u rezultatu predviđanja ukoliko se mijenja razina neujednačenosti. Test je pokazao da za svaku metodu strojnog učenja postoji značajna razlika među grupama rezultata izraženih mjerom GM. Detaljniji pregled rezultata ovog testa prikazan je u tablicama P.5, P.6, P.7 i P.8 u prilogu P2. Ovdje je bitno naglasiti da su testom analizirani veliki uzorci, odnosno grupe rezultata izražene mjerom GM koje imaju više od 30 članova. U takvim slučajevima smanjeno je rasipanje podataka, povećana je snaga testa i nul-hipoteza može biti opovrgnuta čak i ako zapravo postoje vrlo male razlike između grupa rezultata [128].



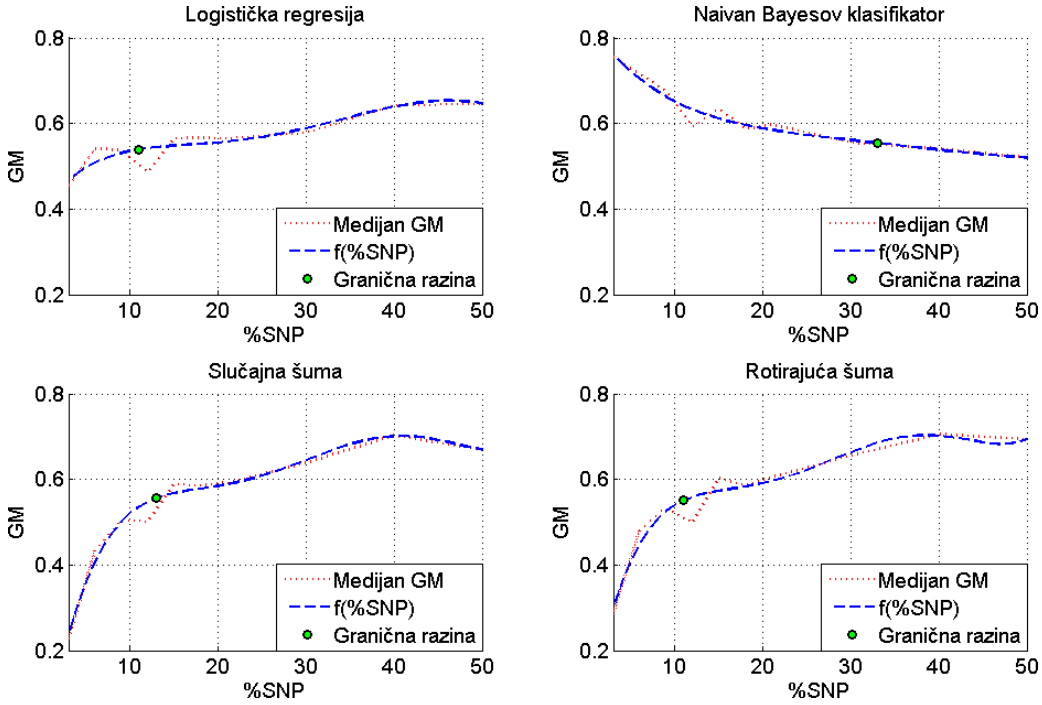
Slika 6.7: Izračun graničnih vrijednosti neujednačenosti podataka za projekt JDT

Posljednji koraci predložene metode služe za otkrivanje točke koju se proglašava graničnom razinom neujednačenosti skupova podataka za pojedinu metodu strojnog učenja. Slike 6.7 i 6.8 prikazuju rezultat te metode za projekte JDT i PDE, a slika 6.9 za oba projekta zajedno. Budući da grupe rezultata GM nisu normalno distribuirane, opisana je ovisnost medijana mjere GM s obzirom na udio manjinske klase (%SNP). Ona je na slikama prikazana crvenom točkastom linijom. Njezina aproksimacija polinomom petog reda $f(\%SNP)$ prikazana je plavom isprekidanom linijom. Potom su izračunate prva derivacija $f'(\%SNP)$ i druga derivacija $f''(\%SNP)$ kako bi se moglo odrediti Arrow-Prattovu mjeru $A(\%SNP) = -f''(\%SNP)/f'(\%SNP)$. Određena je i maksimalna vrijednost Arrow-Prattove mjere $A(\%SNP)$ u monotono rastućem području za metode logističke regresije, slučajne šume i rotirajuće šume, odnosno u monotono padajućem području za metodu naivnog Bayesovog klasifikatora. Na taj način pronađena je granična razina neujednačenosti, koja je na slikama 6.7, 6.8 i 6.9 prikazana zelenom točkom.

Točni iznosi mjere GM i udjela manjinske klase (%SNP) u kojem su pronađene granične razine neujednačenosti prikazani su i u tablici 6.3. Interesantno je uočiti kako za graničnu razinu neujednačenosti mjera GM ima vrijednost otprilike 0.55 u svim slučajevima. Logistička regresija ima vrijednost mjere GM u rasponu [0.53, 0.59], naivan Bayesov klasifikator u rasponu [0.55, 0.61], slučajna šuma u rasponu [0.54, 0.58] te rotirajuća šuma u rasponu [0.55, 0.56]. Također je interesantno uočiti kako metoda rotirajuća šuma ima graničnu razinu neujednačenosti



Slika 6.8: Izračun graničnih vrijednosti neujednačenosti podataka za projekt PDE



Slika 6.9: Izračun graničnih vrijednosti neujednačenosti podataka za projekte JDT i PDE

uvijek za %SNP = 11%. Preostale metode imaju malo širi raspon granične razine neujednačenosti - slučajna šuma u rasponu od 10% do 16%, logistička regresija od 8% do 21% te naivan Bayesov klasifikator od 21% do 33%.

Tablica 6.3: Granična razina neujednačenosti $GM(\%SNP)$ dobivena Arrow-Prattovom mjerom

projekti:	JDT	PDE	JDT + PDE
Logistička regresija	0.53 (8%)	0.59 (21%)	0.54 (11%)
Naivan Bayesov klasifikator	0.55 (21%)	0.61 (30%)	0.55 (33%)
Slučajna šuma	0.54 (10%)	0.58 (16%)	0.55 (13%)
Rotirajuća šuma	0.56 (11%)	0.55 (11%)	0.55 (11%)

6.2.2 Usporedba metoda strojnog učenja

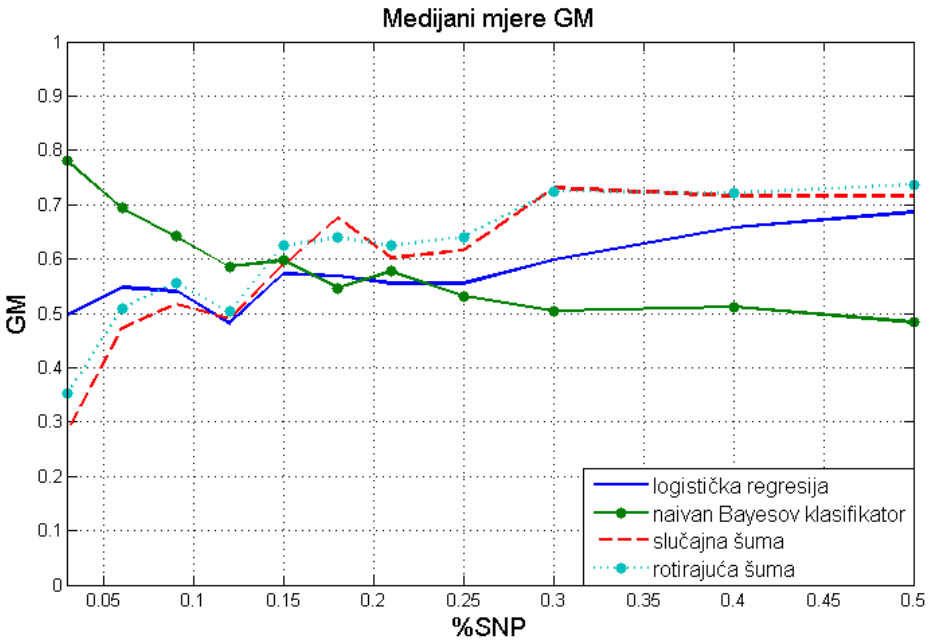
Analiza s ciljem usporedbe metoda strojnog učenja s obzirom na razinu neujednačenost skupova podataka podrazumijeva provedbu statističkog testa značajnosti. Budući da je utvrđeno kako grupe rezultata mjere GM ne prate normalnu distribuciju, koristi se Friedmanov neparametarski test za zavisne varijable, uz razinu značajnosti $\alpha = 0.05$. Broj uzoraka u svakoj grupi rezultata je jednak za svaku metodu strojnog učenja, kao što je prikazano u tablici 6.2. Nul-hipoteza ovog testa je da ne postoji značajna razlika između grupa rezultata izraženih mjerom GM, koji su grupirani s obzirom na korištenu metodu strojnog učenja u određenoj razini neujednačenosti skupova podataka. Drugim riječima, nul-hipoteza je da, za određenu razinu neujednačenosti, ne postoji značajna razlika u rezultatu predviđanja među metodama strojnog učenja. Kao što je bilo objašnjeno u poglavlju 6.1.2, potom je korištena *post hoc* analiza, koja uspoređuje sve parove rezultata korištenih metoda strojnog učenja. Korištena je Holm-Bonferronijeva korekcija p-vrijednosti kako bi se kontrolirala *pogreška na razini skupova usporedbi* (eng. familywise error rate). Nakon *post hoc* analize, određen je rang za svaku od metoda strojnog učenja.

Tablica 6.4 prikazuje rang i srednju vrijednost za svaku metodu strojnog učenja, s obzirom na razine neujednačenosti skupova podataka dobivenih *post hoc* analizom. Tablica je horizontalno podijeljena u 3 dijela. Jedan dio je za skupove podataka iz projekta JDT, drugi za projekt PDE i treći za projekte JDT i PDE zajedno. Za svaku razinu neujednačenosti, proveden je jedan statistički Friedmanov test. Dobivene hi-kvadrat vrijednosti (χ^2) i p-vrijednosti prikazane su u prvom retku svakog dijela tablice. Rang metoda strojnog učenja za svaku razinu neujednačenosti prikazan je u idućim recima. Kod svih testova p-vrijednost je mnogo manja od 0.05, odnosno 0.01. Iz toga se zaključuje da postoje značajne razlike među metodama strojnog učenja za svaku promatranu razinu neujednačenosti, čime je provedba rangiranja opravdana. Ponovo je važno naglasiti kako veličina uzoraka analiziranih testom uzrokuje smanjeno rasipanje po-

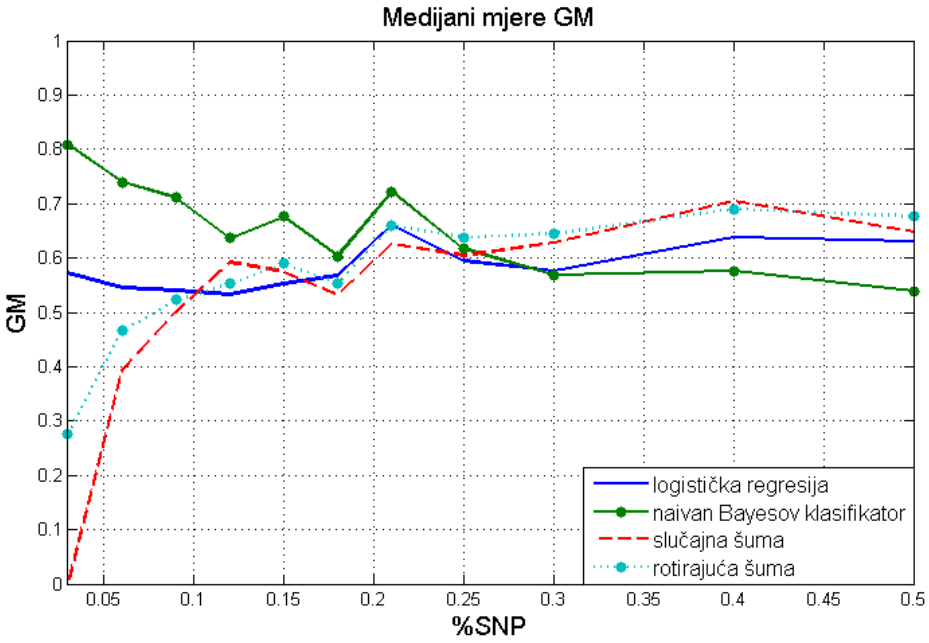
Tablica 6.4: Rang i (srednja vrijednost) mjere GM za metode strojnog učenja s obzirom u različitim razinama neujednačenosti

Granice %SNP:	0% - 3%	3% - 6%	6% - 9%	9% - 12%	12% - 15%	15% - 18%	18% - 21%	21% - 25%	25% - 30%	30% - 40%	40% - 50%
projekt:	**JDT**										
χ^2	5934.68	2356.50	777.05	649.94	111.57	283.29	278.80	299.23	162.98	650.89	171.98
p-vrijednost	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Logistička regresija	2 (0.43)	2 (0.54)	2 (0.53)	3 (0.47)	3 (0.57)	3 (0.56)	4 (0.53)	3 (0.56)	3 (0.59)	3 (0.64)	3 (0.68)
Naivan Bayesov klasifikator	**1 (0.78)**	**1 (0.69)**	**1 (0.64)**	**1 (0.58)**	2 (0.59)	3 (0.54)	3 (0.55)	3 (0.53)	4 (0.50)	4 (0.51)	4 (0.48)
Slučajna šuma	4 (0.23)	4 (0.47)	4 (0.51)	2 (0.48)	2 (0.59)	**1 (0.67)**	**1 (0.58)**	2 (0.61)	**1 (0.73)**	**1 (0.70)**	2 (0.71)
Rotirajuća šuma	3 (0.30)	3 (0.50)	2 (0.54)	2 (0.48)	**1 (0.62)**	2 (0.63)	**1 (0.59)**	**1 (0.64)**	**1 (0.72)**	**1 (0.71)**	**1 (0.74)**
projekt:	**PDE**										
χ^2	4314.19	1972.24	897.86	105.25	379.16	180.16	91.64	37.73	241.04	426.55	301.89
p-vrijednost	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Logistička regresija	2 (0.47)	2 (0.53)	2 (0.54)	3 (0.53)	4 (0.55)	2 (0.56)	2 (0.66)	2 (0.60)	3 (0.58)	3 (0.64)	2 (0.62)
Naivan Bayesov klasifikator	**1 (0.73)**	**1 (0.74)**	**1 (0.71)**	**1 (0.64)**	**1 (0.66)**	**1 (0.60)**	**1 (0.72)**	2 (0.61)	3 (0.57)	4 (0.58)	4 (0.57)
Slučajna šuma	4 (0.21)	4 (0.37)	4 (0.49)	2 (0.59)	2 (0.58)	2 (0.54)	3 (0.62)	2 (0.61)	**1 (0.63)**	**1 (0.70)**	2 (0.64)
Rotirajuća šuma	3 (0.27)	3 (0.44)	2 (0.52)	3 (0.55)	2 (0.58)	2 (0.56)	2 (0.65)	**1 (0.64)**	**1 (0.64)**	**1 (0.69)**	**1 (0.67)**
projekti:	**JDT + PDE**										
χ^2	10175.03	4310.62	1651.86	737.63	341.32	41.73	172.97	277.75	392.45	1067.06	471.00
p-vrijednost	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$	$p < 0.01$
Logistička regresija	2 (0.45)	2 (0.54)	2 (0.54)	4 (0.48)	4 (0.56)	4 (0.56)	4 (0.56)	3 (0.57)	3 (0.58)	3 (0.64)	3 (0.64)
Naivan Bayesov klasifikator	**1 (0.75)**	**1 (0.71)**	**1 (0.68)**	**1 (0.59)**	**1 (0.63)**	**1 (0.59)**	2 (0.59)	3 (0.56)	4 (0.54)	4 (0.54)	4 (0.52)
Slučajna šuma	4 (0.23)	4 (0.43)	4 (0.50)	2 (0.50)	3 (0.59)	**1 (0.59)**	2 (0.59)	2 (0.61)	**1 (0.66)**	**1 (0.70)**	2 (0.67)
Rotirajuća šuma	3 (0.29)	3 (0.48)	2 (0.53)	2 (0.50)	2 (0.60)	**1 (0.59)**	**1 (0.61)**	**1 (0.64)**	**1 (0.67)**	**1 (0.70)**	**1 (0.69)**

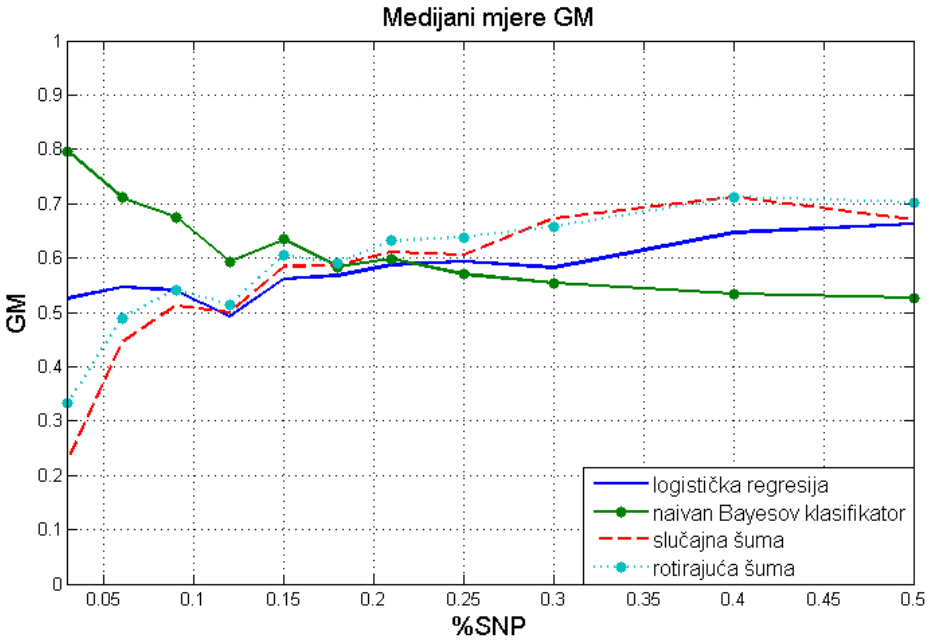
dataka i povećanu snagu testa [128]. Zbog toga nul-hipoteza može biti opovrgnuta čak i ako zapravo postoje vrlo male razlike između grupa rezultata [128]. Radi boljeg uvida u rezultate rangiranja, slike 6.10, 6.11 i 6.12 usporedno prikazuju medijan mjere GM dobivene odabranim metodama strojnog učenja za svaku razinu neujednačenosti.



Slika 6.10: Usporedni prikaz rezultata primjenjenih metoda strojnog učenja za projekt JDT



Slika 6.11: Usporedni prikaz rezultata primjenjenih metoda strojnog učenja za projekt PDE



Slika 6.12: Usporedni prikaz rezultata primjenjenih metoda strojnog učenja za projekte JDT i PDE

6.3 Diskusija rezultata

Predstavljeno empirijsko istraživanje provelo je opsežnu usporedbu 5 metoda strojnog učenja u uvjetima različite razine neujednačenosti skupova podataka za SDP. Odabrani skupovi podataka prikupljeni su iz 11 inačica projekta Eclipse JDT i 10 inačica projekta Eclipse PDE. Budući da slična usporedba do sada nije provedena u srodnoj literaturi, predložena je metoda za provjeru utjecaja razine neujednačenosti skupova podataka na performanse metoda strojnog učenja. Utjecaj neujednačenosti promatrao se kroz otkrivanje granične razine neujednačenosti, kod koje se performanse odabrane metode strojnog počinju mijenjati i ulaze u područje nestabilnosti. Drugi aspekt utjecaja neujednačenosti bio je otkrivanje odgovarajuće metode strojnog učenja za svaku od razina neujednačenosti.

Analizirane su sljedeće metode strojnog učenja: logistička regresija, naivan Bayesov klasifikator, metoda potpornih vektora, slučajna šuma i rotirajuća šuma. Metoda potpornih vektora postigla je jako loše rezultate te oni nisu bili prikazani u prethodnom poglavlju. Ustanovljeno je da rezultati te metode nisu bolji od nasumičnog pogađanja (AUC je iznosio gotovo uvijek 0.5), značajnije su lošiji od svih ostalih metoda i samo bi unosili šum u provedenim analizama. Performanse odabranih metoda vrednovane su mjerom geometrijske sredine (GM), budući da je to mjera koja pruža najbolju informaciju o uspješnosti otkrivanja pripadnika i većinske i manjinske klase [129, 130].

Granična razina neujednačenosti

Granična razina neujednačenosti skupova podataka od koje dolazi do promjene u performansama različita je za svaku metodu strojnog učenja. Međutim, svim metodama je zajedničko da kod najviših razina neujednačenosti ulaze u područje najveće nestabilnosti. To je uočljivo iz raspona rezultata, koji su prikazani u pravokutnim dijagramima prethodnog poglavlja. Dok za modele temeljene na metodama logističke regresije, slučajne šume i rotirajuće šume performanse opadaju s daljnjim porastom neujednačenosti, za metodu naivnog Bayesovog klasifikatora one počinju rasti. Takav trend nije pronađen u srodnim istraživačkim radovima. Mogući uzrok ovakvog ponašanja potražen je unutar samih algoritama. Jedino naivan Bayesov klasifikator uzima u obzir udio manjinske klase unutar parametra $P(C_i)$ iz izraza 2.6. Suprotan trend u performansama naivnog Bayesovog klasifikatora iziskuje i drugačiju interpretaciju granične razine neujednačenosti. Performanse ove metode počinju opadati u uvjetima manje razine neujednačenosti.

Za različite metode strojnog učenja postoje manje razlike u udjelu manjinske klase %SNP, kod koje je otkrivena granična razine neujednačenosti. Međutim, sve metode strojnog učenja imaju približno jednaku vrijednosti mjere GM, kod koje nastupa granična razina neujednačenosti. Stoga, moguće je dati odgovor i na prvo istraživačko pitanje. Za sve metode može se očekivati promjena performansi kod vrijednosti mjere GM koja iznosi oko 55%. Rotirajuća šuma tu točku dostiže kada manjinska klasa padne na otprilike 11%, slučajna šuma i logistička regresija između 10% i 13%, a naivan Bayesov klasifikator između 20% i 33%.

Usporedbom rezultata prikazanih pravokutnim dijagramima i izračunom graničnih razina neujednačenosti otkrivene su određene konzistentnosti. To ukazuje da predložena metoda za utvrđivanje granične razine neujednačenosti daje korisne rezultate. U uvjetima veće razine neujednačenosti uočava se opadanje medijana i srednje vrijednosti te porast raspona rezultata izraženih u mjeri GM. Ukupni raspon rezultata počinje se širiti na gotovo cijelo područje u kojem su rezultati mogući [0, 1]. To ukazuje na iznimno nestabilno ponašanje metoda strojnog učenja. Interkvartilni raspon se širi nešto sporije, ali i on doseže velike vrijednosti za najnižu razinu neujednačenosti, ispod 3%. Primjerice, granična razina neujednačenosti za rotirajuću šumu iznosi 11% u svim slučajevima. Porastom neujednačenosti iza te vrijednosti, ukupni raspon mjere GM poprima gotovo dvostruko veće vrijednosti. To znači da se granična razina neujednačenosti preklapa s točkom u kojoj počinje nestabilnije ponašanje ove metode strojnog učenja. S druge strane, za naivan Bayesov klasifikator se isto ne može uočiti. Ta metoda je

jedina među odabranim metodama strojnog učenja koja ima bolje rezultate u uvjetima više razine neujednačenosti. Međutim, i ova metoda ulazi u područje velikih odstupanja (ukupni raspon od 0 do 1) za najviše razine neujednačenosti, ispod 3%.

Usporedba metoda strojnog učenja

Odabir najbolje metode strojnog učenja može biti vođen razinom neujednačenosti skupova podataka. Iz rezultata se može uočiti da su različite metode strojnog učenja pogodne za različite razine neujednačenosti. U većini slučajeva, najbolji model predviđanja je rotirajuća šuma, odnosno naivan Bayesov klasifikator. Najmanja odstupanja performansi mogu se uočiti za logističku regresiju. Ta metoda je rijetko najmanje uspješna, a u razinama visoke neujednačenosti, kada udio manjinske klase padne ispod 10%, ona je drugi najbolji algoritam. Također, može se uočiti da neki algoritmi ipak jesu najbolji u određenim razinama neujednačenosti, bez obzira iz kojeg projekta podaci potječu. To daje odgovor i na drugo istraživačko pitanje postavljeno u uvodu ovog poglavlja. Grupna metoda rotirajuće šume je najuspješniji algoritam za klasifikaciju u ujednačenim skupovima i umjereno neujednačenim skupovima kada udio manjinske klase ne pada ispod 20%. Grupna metoda slučajne šume ima slične performanse, ali gotovo nikada značajno bolje. Kada udio manjinske klase padne ispod 20%, naivan Bayesov klasifikator postaje najuspješniji algoritam za klasifikaciju. Ta metoda postaje neprikosnovena za razine visoke razine neujednačenosti, ispod 10%.

Interesantno je uočiti da promjene najuspješnijeg algoritma dolaze u blizini vrijednosti mjere GM kod koje nastupa granična razina neujednačenosti. Za razinu neujednačenosti koja je viša od granične, performanse naivnog Bayesovog klasifikatora počinju biti bolje. Kada mjera GM dosegne vrijednost oko 0.6, ova metoda postaje najuspješnija. Upravo oko te vrijednosti, metode rotirajuća šuma i slučajna šuma istovremeno dolaze do vlastite granične razine neujednačenosti i prestaju biti značajno bolje od ostalih metoda. Ova pravilnost daje još jednu potvrdu važnosti otkrivanja graničnih razina neujednačenosti za sve metode strojnog učenja.

Valjanost istraživanja

Valjanost konstrukcije je narušena jer nisu korišteni industrijski podaci i postoji problem neispravnosti kojima nije pronađena veza s odgovarajućom predajom programskog koda. Međutim, koriste se podaci iz projekata koji po veličini, složenosti i dugovječnosti što više nalikuju na industrijske projekte. Iako su podaci i metode strojnog učenja pomno odabrani, vanjska valjanost

je ipak ograničena. Dakle, dobiveni zaključci su reprezentativni u onoj mjeri u kojoj su reprezentativni korišteni podaci i metode strojnog učenja. Unutarnja valjanost je ograničena na proučavanje isključivo utjecaja neujednačenosti skupova podataka. Međutim, osim neujednačenosti skupova podataka, moguće je da i drugi čimbenici utječu na performanse metoda strojnog učenja. To mogu biti čimbenici poput ranije spomenutih stopa povezivanja ili pak razine kvalitete razvojnih procesa koja napreduje s evolucijom projekata. Takve čimbenike teško je razlučiti i odvojiti, te je stoga i korištena metoda analize slučaja. Za postizanje veće snage zaključaka, potrebno je ponoviti studije nad projektima iz drugih razvojnih zajednica i domena primjene.

Poglavlje 7

Zaključak

U razvoju složenih programskih sustava, velika sredstva ulažu se u verifikaciju programskog koda. Strategija verifikacije je pritom vrlo važan korak, koji treba usmjeriti sredstva u problematične programske module. Poželjno je što ranije započeti s provedbom verifikacije kako bi se povećala učinkovitost testiranja. Uspješno predviđanje programskih neispravnosti može pomoći u definiranju strategije verifikacije programskih modula, zasnovane na kvarovima. To se postiže primjenom odgovarajuće metode strojnog učenja na povijesnim podacima. U evoluciji složenih programskih sustava, analitički model predviđanja izgradi se na podacima iz prethodne inačice i može se početi primjenjivati već od najranijih faza razvoja iduće inačice. Uz rastuću složenost programskih sustava te njihovu primjenu u svim područjima života i rada, od kojih su neka iznimno osjetljiva i na najmanju pogrešku, predviđanje programskih neispravnosti ima zajamčeno važnu ulogu.

Ova disertacija usmjerena je na unapređenje dvaju problematičnih aspekata u predviđanju programskih neispravnosti. Uočena je nemogućnost usporedbe i poopćenja zaključaka iz provedenih analiza slučaja. Nedovoljna valjanost konstrukcije i zaključaka često je uzrok tog problema. Nedostatak normi u načinu provedbe prikupljanja podataka pritom je najveći izazov kojim se bavi ova disertacija. Istovremeno, uočen je velik napor istraživačke zajednice u pronalasku najbolje metode strojnog učenja, zasad bez značajnijeg uspjeha. Fokus ove disertacije stoga je na pronalasku opisa okruženja u kojem se primjenjuje odabrane metode strojnog učenja za predviđanje programskih neispravnosti.

Unapređenje postupaka prikupljanja podataka

U rješavanju prvog problema istražene su norme i postojeći postupci prikupljanja podataka. Za proces praćenja zahtjeva, odnosno neispravnosti, postoji norma IEEE 1044-2009. S druge strane, za proces upravljanja inačicama, odnosno izmjena programskog koda, norma još ne postoji. Izostanak normi otežava povezivanje podataka iz ta dva razvojna procesa, te su stoga razvijeni postupci kojima se zaobilazi taj problem. Stoga, buduća norma svakako bi trebala omogućiti nedvosmisleno povezivanje programskog koda i neispravnosti jer podaci koji se tako dobiju imaju jako veliku vrijednost za unapređenje postupaka razvoja složenih programskih struktura. Pronađeni su svi parametri prikupljanja podataka po kojima se ti postupci razlikuju ili kojima nisu pristupili dovoljno precizno. Temeljem tih spoznaja definirana je procedura za prikupljanje podataka za predviđanje programskih neispravnosti, čiji je cilj povećati prikladnost podataka. Nastavno na definiranu proceduru, istražene su i tehnike povezivanja podataka iz razvojnih repozitorija, među kojima ne postoji formalna veza, a neophodne su za prikupljanje podataka. Izgrađena je i vlastita tehnika, zasnovana na uporabi regularnih izraza. Predložen je okvir opsežne komparativne studije, čijom su provedbom vrednovane postojeće procedure prikupljanja podataka i tehnike povezivanja razvojnih repozitorija, uključujući i vlastite. Rezultati te studije pokazali su da predložena procedura prikupljanja podataka i tehnika povezivanja razvojnih repozitorija daju najtočnije podatke. Nova tehnika i procedura čine temelj sustavnog prikupljanja podataka za predviđanje programskih neispravnosti te su implementirane u alat Bug-Code Analyzer (BuCo). Alatom BuCo omogućeno je konzistentno prikupljanje podataka kroz algoritam za prikupljanje podataka iz nestrukturiranih i formalno nepovezanih repozitorija programskog koda i neispravnosti. Time su ostvareni prvi i drugi izvorni znanstveni doprinosi ove disertacije. Prednosti opisanih unapređenja je veća točnost prikupljenih podataka i procedura, koja ne ostavlja parametre prikupljanja podataka otvorenima za interpretaciju.

Unapređenje postupaka izgradnje modela predviđanja

Za rješavanje drugog problema, istraženo je ponašanje metoda strojnog učenja, s obzirom na razinu neujednačenosti skupova podataka. Neujednačenost skupova podataka je inherentno svojstvo u predviđanju programskih neispravnosti, a prepoznato je kao zajednički problem i u drugim područjima primjene metoda strojnog učenja. Ovom disertacijom analiziran je utjecaj razine neujednačenosti skupova podataka na izgradnju modela predviđanja. Predložena je metoda za utvrđivanje granične razine neujednačenosti od koje dolazi do negativne promjene u

performansama metoda strojnog učenja. Ona se temelji na uporabi Arrow-Prattove mjere, čija se izvorna namjena odnosila na procjenu sklonosti izbjegavanja rizika u ekonomiji. Također, predložena je metoda za usporedbu metoda strojnog učenja s obzirom na razinu neujednačenosti skupova podataka. Ona se temelji na uporabi statističkih testova značajnosti. Potom je provedeno empirijsko istraživanje nad sustavno prikupljenim podacima iz velikog broja uzastopnih inačica projekata otvorenog koda. Korištene su pouzdane metode strojnog učenja, koje su u srodnoj literaturi ostvarivale izvrsne rezultate. Rezultati empirijskog istraživanja potvrdili su podudaranje izračunatih graničnih vrijednosti neujednačenosti i negativnih promjena u performansama metoda strojnog učenja. Time je ostvaren i treći izvorni znanstveni doprinos ove disertacije. Također, uočena je i konzistentnost u odabiru odgovarajuće metode strojnog učenja, s obzirom na razinu neujednačenosti skupova podataka. Time je ustanovljen postupak za ostvarivanje četvrtog izvornog znanstvenog doprinosa ove disertacije. Plan budućeg istraživanja je provesti opsežnu empirijsku studiju usporedbe kombinacija metoda za predviđanje programskih neispravnosti iz skupa metoda za pripremu podataka i metoda za strojno učenje.

Valjanost istraživanja i zaključaka

Ovo poglavlje komentira valjanost provedenog istraživanja sukladno preporukama iz poglavlja 1.3. Provedene analize slučaja koristile su javno dostupne podatke, što dovodi do lakše ponovljivosti rezultata, odnosno veće valjanosti zaključaka. Valjanost konstrukcije narušena je jer nisu korišteni industrijski podaci, a vanjska valjanost provedenog istraživanja ograničena je na odabrane projekte. Riječ je o projektima otvorenog koda, čija su domena primjene razvojni, mrežni i poslovni sustavi. Zaključci su, stoga, reprezentativni u onoj mjeri u kojoj su korišteni podaci reprezentativni za domenu primjene predviđanja programskih neispravnosti. Snaga zaključaka može se značajno unaprijediti ukoliko se istraživanje provede i s industrijskim podacima i projektima drugačijih domena primjene. Ovaj nedostatak je utoliko manji jer su odabrani projekti koji po veličini i složenosti što više nalikuju na industrijske. Odabir metoda strojnog učenja također ograničava vanjsku valjanost provedenog istraživanja. Ispravnost podataka utječe na valjanost zaključaka ove disertacije. Primjenom predloženog algoritma za povezivanje razvojnih repozitorija i procedure za prikupljanje podataka dobivaju se podaci veće točnosti od prethodnih studija slučaja u srodnoj literaturi. Međutim, podaci nisu apsolutno točni zbog veza koje nedostaju između ispravljenih neispravnosti i programskog koda. U dijelu disertacije koji promatra utjecaj neujednačenosti skupova podataka, ograničena je i unutarnja

valjanost jer su mogući utjecaji i drugih čimbenika. Budući da čimbenike poput ranije spomenute točnosti podataka, koja je narušena vezama koje nedostaju, ili razine kvalitete razvojnih procesa, koja napreduje s evolucijom projekata, nije moguće odvojiti, opravdana je uporaba metode analize slučaja.

Korist i primjena postignutih zaključaka

Vrijednost ove disertacije je da će ustanovljeni postupak prikupljanja podataka, razvijeni algoritam za povezivanje podataka i predložene metode za određivanje utjecaja razine neujednačenosti biti od iznimne važnosti za buduća istraživanja. Spoznaje ove disertacije mogu pomoći u procjeni valjanosti konstrukcije za buduća istraživanja, odnosno rizika za valjanost zaključaka koji proizlazi iz nedovoljno definirane procedure prikupljanja podataka ili uporabe neprikladne metode strojnog učenja. Prikupljanje podataka je i dalje vremenski iznimno zahtjevan proces, ali primjena alata BuCo olakšava taj proces te njegovu provedbu čini konzistentnom. To će omogućiti širenje istraživanja na druga razvojna okruženja te poopćenje dobivenih zaključaka. Provedena unapređenja otvorila su i nov pogled na problem odabira odgovarajuće metode strojnog učenja. Iako i dalje nedostaje način kako karakterizirati domenu primjene, predložen je način za karakterizaciju okruženja primjene putem razine neujednačenosti u skupovima podataka. Promatranjem performansi metoda strojnog učenja u okruženju različite razine neujednačenosti skupova podataka uočene su određene pravilnosti. Ta spoznaja omogućuje opis okruženja primjene koji stručnjacima osiguranja kvalitete može poslužiti kao smjernica za provedbu predviđanja programskih neispravnosti.

Primjena zaključaka ove disertacije odnosi se na donošenje strategije verifikacije programskog koda za složene programske sustave u evoluciji. Unapređenje postupaka za testiranje programskog koda zasnovanog na kvarovima omogućuje pronalazak odgovarajućeg analitičkog modela predviđanja neispravnosti. Također, omogućuje i provedbu strategije verifikacije već u najranijim fazama razvoja, što može dovesti do ranijeg otkrivanja neispravnosti i veće učinkovitosti testiranja, odnosno smanjenja troškova.

Literatura

[1] Stecklein, J. M., Dabney, B., Jimand Dick, Haskins, R., Billand Lovell, Moroney, G., "Error cost escalation through the project life cycle", in International Council on Systems Engineering (INCOSE) Foundation. NASA, 2004, str. 1-11.

[2] Fenton, N. E., Ohlsson, N., "Quantitative analysis of faults and failures in a complex software system", IEEE Trans. Softw. Eng., Vol. 26, No. 8, 2000, str. 797–814.

[3] Galinac Grbac, T., Runeson, P., Huljenić, D., "A second replicated quantitative analysis of fault distributions in complex software systems", IEEE Trans. Softw. Eng., Vol. 39, No. 4, Apr. 2013, str. 462–476.

[4] Andersson, C., Runeson, P., "A replicated quantitative analysis of fault distributions in complex software systems", IEEE Trans. Softw. Eng., Vol. 33, No. 5, 2007, str. 273–286.

[5] Tedre, M., The Science of Computing: Shaping a Discipline. Chapman & Hall/CRC, 2014.

[6] Boehm, B. W., Software Engineering Economics, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981.

[7] Boehm, B., Basili, V. R., "Software defect reduction top 10 list", Computer, Vol. 34, No. 1, Jan. 2001, str. 135–137.

[8] Iso/ie, C., "Iso/iec 12207:2008: Systems and software engineering - software life cycle processes",
urlhttp://www.iso.org, Feb. 2008.

[9] Vliet, H. v., Software Engineering: Principles and Practice, 3rd ed. Wiley Publishing, 2008.

[10] "IEEE Standard Classification for Software Anomalies", IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993), Jan 2010, str. 1-23.

[11] Pfleeger, S. L., Software Engineering: Theory and Practice, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[12] "Software and systems engineering software testing part 2:test processes", ISO/IEC/IEEE 29119-2:2013(E), Sept 2013, str. 1-68.

[13] Tian, J., Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement. Wiley, 2005.

[14] "IEEE Standard for System and Software Verification and Validation", IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004), May 2012, str. 1-223.

[15] Musa, J. D., "Operational profiles in software-reliability engineering", IEEE Software, Vol. 10, No. 2, March 1993, str. 14-32.

[16] Koziolek, H., "Operational profiles for software reliability", in Dependability Engineering, ser. Trustworthy Software Systems, Hasselbring, W., Giesecke, S., (ur.). GITO-Verlag, Berlin, 2006, September 2005, Vol. 2, ch. 6, str. 119–142.

[17] Whittaker, J. A., Voas, J., "Toward a more reliable theory of software reliability", Computer, Vol. 33, No. 12, Dec. 2000, str. 36–42.

[18] Whittaker, J. A., Poore, J. H., "Markov analysis of software specifications", ACM Trans. Softw. Eng. Methodol., Vol. 2, No. 1, Jan. 1993, str. 93–106.

[19] Shukla, R., Carrington, D., Strooper, P., "Systematic operational profile development for software components", in Software Engineering Conference, 2004. 11th Asia-Pacific, Nov 2004, str. 528-537.

[20] Galinac Grbac, T., Huljenić, D., "On the probability distribution of faults in complex software systems", Information and Software Technology, Vol. 58, 2015, str. 250 - 258.

[21] Robillard, M., Walker, R., Zimmermann, T., "Recommendation systems for software engineering", IEEE Software, Vol. 27, No. 4, 2010, str. 80-86.

[22] Zhang, H., “On the distribution of software faults.”, IEEE Trans. Software Eng., Vol. 34, No. 2, 2008, str. 301-302.

[23] Concas, G., Marchesi, M., Murgia, A., Tonelli, R., Turnu, I., “On the distribution of bugs in the eclipse system.”, IEEE Trans. Software Eng., Vol. 37, No. 6, 2011, str. 872-877.

[24] Borg, M., Runeson, P., Ardö, A., “Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability”, Vol. 19, No. 6, 2014, str. 1565–1616.

[25] Naur, P., Randell, B., (ur.), Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO, 1969.

[26] Fenton, N. E., Neil, M., “A critique of software defect prediction models”, IEEE Trans. Softw. Eng., Vol. 25, No. 5, 1999, str. 675–689.

[27] Leng, J., Sharrock, W., Handbook of Research on Computational Science and Engineering: Theory and Practice. Engineering Science Reference, 2012, Vol. 2.

[28] Runeson, P., Höst, M., “Guidelines for conducting and reporting case study research in software engineering”, Empirical Softw. Engg., Vol. 14, No. 2, Apr. 2009, str. 131–164.

[29] Robson, C., Real World Research - A Resource for Social Scientists and Practitioner-Researchers, 2nd ed. Malden: Blackwell Publishing, 2002.

[30] Kitchenham, B., “Procedures for performing systematic reviews”, 2004.

[31] Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S., “Systematic literature reviews in software engineering - a systematic literature review”, Inf. Softw. Technol., Vol. 51, No. 1, Jan. 2009, str. 7–15.

[32] Runeson, P., Host, M., Rainer, A., Regnell, B., Case Study Research in Software Engineering: Guidelines and Examples, 1st ed. Wiley Publishing, 2012.

[33] Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., “Data collection for software defect prediction – an exploratory case study of open source software projects”, in Proceedings of MIPRO ’14, Opatija, Croatia, 2015, str. 513–519.

[34] Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., "Software defect prediction with bug-code analyzer - a data collection tool demo", in Proc. of SoftCOM '14, 2014.

[35] Mauša, G., Perković, P., Galinac Grbac, T., Štajduhar, I., "Techniques for bug-code linking", in Proc. of SQAMIA '14, 2014, str. 47–55.

[36] Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., "A systemathic data collection procedure for software defect prediction", Vol. 12, No. 4, 2015, str. to be published.

[37] Galinac Grbac, T., Mauša, G., Dalbelo Bašić, B., "Stability of software defect prediction in relation to levels of data imbalance.", in Proceedings of SQAMIA '13, Novi Sad, Serbia, 2013, str. 1-10.

[38] Mauša, G., Bogunović, N., Grbac Galinac, T., Dalbelo Bašić, B., "Rotation forest in software defect prediction.", in SQAMIA, ser. CEUR Workshop Proceedings, Budimac, Z., Hericko, M., (ur.), Vol. 1375. CEUR-WS.org, 2015, str. 35-43.

[39] Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S., "A systematic literature review on fault prediction performance in software engineering", IEEE Trans. Softw. Eng., Vol. 38, No. 6, 2012, str. 1276–1304.

[40] Runeson, P., Ohlsson, M. C., Wohlin, C., "A classification scheme for studies on fault-prone components", in Product Focused Software Process Improvement, Third International Conference, PROFES 2001, Kaiserslautern, Germany, September 10-13, 2001, Proceedings, 2001, str. 341–355.

[41] Myrtveit, I., Stensrud, E., Shepperd, M., "Reliability and validity in comparative studies of software prediction models", Software Engineering, IEEE Transactions on, Vol. 31, No. 5, May 2005, str. 380-391.

[42] Shepperd, M. J., Kadoda, G. F., "Comparing software prediction techniques using simulation.", IEEE Trans. Software Eng., Vol. 27, No. 11, 2001, str. 1014-1022.

[43] Menzies, T., Shepperd, M., "Special issue on repeatable results in software engineering prediction", Empirical Software Engineering, Vol. 17, No. 1-2, 2012, str. 1–17.

[44] Lessmann, S., Baesens, B., Mues, C., Pietsch, S., "Benchmarking classification models for software defect prediction: A proposed framework and novel findings", IEEE Trans. Softw. Eng., Vol. 34, No. 4, Jul. 2008, str. 485–496.

[45] D'Ambros, M., Lanza, M., Robbes, R., "Evaluating defect prediction approaches: A benchmark and an extensive comparison", Empirical Softw. Engg., Vol. 17, No. 4-5, 2012, str. 531–577.

[46] Kelman, C. W., Bass, J., Holman, D., "Research use of linked health data—a best practice protocol", Australian NZ J. Public Health, Vol. 26, 2002, str. 251–255.

[47] Gill, L., Methods for automatic record matching and linking and their use in national statistics. Oxford University, 2001.

[48] Ali, N., Guéhéneuc, Y.-G., Antoniol, G., "Trustrace: Mining software repositories to improve the accuracy of requirement traceability links.", IEEE Trans. Software Eng., Vol. 39, No. 5, 2013, str. 725-741.

[49] Śliwerski, J., Zimmermann, T., Zeller, A., "When do changes induce fixes?", SIGSOFT Softw. Eng. Notes, Vol. 30, No. 4, May 2005, str. 1–5.

[50] Kim, D., Tao, Y., Kim, S., Zeller, A., "Where should we fix this bug? a two-phase recommendation model", IEEE Trans. Softw. Eng., Vol. 39, No. 11, 2013, str. 1597–1610.

[51] Thomas, S. W., Nagappan, M., Blostein, D., Hassan, A. E., "The impact of classifier configuration and classifier combination on bug localization.", IEEE Trans. Software Eng., Vol. 39, No. 10, 2013, str. 1427-1443.

[52] Zhang, J., Wang, X., Hao, D., Xie, B., Zhang, L., Mei, H., "A survey on bug-report analysis", Science China Information Sciences, Vol. 58, No. 2, 2015, str. 1–24.

[53] González-Barahona, J. M., Robles, G., "On the reproducibility of empirical software engineering studies based on data retrieved from development repositories", Empirical Softw. Engg., Vol. 17, No. 1-2, Feb. 2012, str. 75–89.

[54] Rodríguez, D., Ruiz, R., Riquelme, J. C., Aguilar-Ruiz, J. S., "Searching for rules to detect defective modules: A subgroup discovery approach.", Inf. Sci., Vol. 191, 2012, str. 14-30.

[55] Gao, K., Khoshgoftaar, T. M., "Software defect prediction for high-dimensional and class-imbalanced data.", in SEKE. Knowledge Systems Institute Graduate School, 2011, str. 89-94.

[56] Folleco, A., Khoshgoftaar, T., Napolitano, A., "Comparison of four performance metrics for evaluating sampling techniques for low quality class-imbalanced data", in Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on, 2008, str. 153-158.

[57] Basili, V. R., Weiss, D., "A methodology for collecting valid software engineering data", IEEE Computer Society Trans. Software Engineering, Vol. 10, No. 6, 1984, str. 728-738.

[58] Rodriguez, D., Herraiz, I., Harrison, R., "On software engineering repositories and their open problems", in Proceedings of RAISE '12, 2012, str. 52-56.

[59] Shepperd, M. J., Song, Q., Sun, Z., Mair, C., "Data quality: Some comments on the nasa software defect datasets.", IEEE Trans. Software Eng., Vol. 39, No. 9, 2013, str. 1208-1215.

[60] Bachmann, A., Bernstein, A., "Data retrieval, processing and linking for software process data analysis", Tech. Rep. IFI-2009.0003b, 2009.

[61] Crowston, K., "A coordination theory approach to organizational process design", Organization Science, Vol. 8, No. 2, 1997, str. 157–175.

[62] Bettenburg, N., Just, S., Schroter, A., Weiss, C., Premraj, R., Zimmermann, T., "Quality of bug reports in eclipse", in Proceedings of the 2007 OOPSLA Workshop on Eclipse Technology eXchange. New York, NY, USA: ACM Press, Oct. 2007.

[63] Herzig, K., Just, S., Zeller, A., "It's not a bug, it's a feature: How misclassification impacts bug prediction", in Proceedings of the 2013 International Conference on Software Engineering, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, str. 392–401.

[64] Bosu, A., "Mining repositories to reveal the community structures of open source software projects", in Proceedings of the 50th Annual Southeast Regional Conference, 2012, Tuscaloosa, AL, USA, March 29-31, 2012, 2012, str. 397–398.

[65] Zimmermann, T., Premraj, R., Zeller, A., "Predicting defects for eclipse", in Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on, May 2007, str. 9-9.

[66] Bass, J., Garfield, C., “Statistical linkage keys: How effective are they?”, in Proceedings of the 12th Symposium on Health Data Linkage. Syndey, NSW, Australia: Public Health Information Development Unit, 2002, str. 40–45.

[67] Fellegi, I. P., Sunter, A. B., “A theory for record linkage”, Journal of the American Statistical Association, Vol. 64, 1969, str. 1183–1210.

[68] Śliwerski, J., Zimmermann, T., Zeller, A., “When do changes induce fixes?”, SIGSOFT Softw. Eng. Notes, Vol. 30, No. 4, 2005, str. 1–5.

[69] Jiang, T., Tan, L., Kim, S., “Personalized defect prediction.”, in ASE, 2013, str. 279-289.

[70] Hu, W., Wong, K., “Using citation influence to predict software defects”, in Proceedings of the 10th Working Conference on Mining Software Repositories, ser. MSR ’13, 2013, str. 419–428.

[71] Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K.-i., Adams, B., Hassan, A. E., “Revisiting common bug prediction findings using effort-aware models”, in Proceedings of the 2010 IEEE International Conference on Software Maintenance, ser. ICSM ’10, 2010, str. 1–10.

[72] Bird, C., Nagappan, N., Gall, H., Murphy, B., Devanbu, P. T., “Putting it all together: Using socio-technical networks to predict failures.”, in ISSRE. IEEE Computer Society, 2009, str. 109-119.

[73] Mizuno, O., Ikami, S., Nakaichi, S., Kikuno, T., “Spam filter based approach for finding fault-prone software modules.”, in MSR, 2007, str. 4.

[74] Inozemtseva, L., Hemmati, H., Holmes, R., “Using fault history to improve mutation reduction”, in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2013, 2013, str. 639–642.

[75] Zhang, F., Mockus, A., Keivanloo, I., Zou, Y., “Towards building a universal defect prediction model”, in Proceedings of the 11th Working Conference on Mining Software Repositories, ser. MSR 2014. New York, NY, USA: ACM, 2014, str. 182–191.

[76] Zaman, S., Adams, B., Hassan, A. E., “Security versus performance bugs: A case study on firefox”, in Proceedings of the 8th Working Conference on Mining Software Repositories, ser. MSR ’11. New York, NY, USA: ACM, 2011, str. 93–102.

[77] Shivaji, S., Jr., E. J. W., Akella, R., Kim, S., "Reducing features to improve code change-based bug prediction.", IEEE Trans. Software Eng., Vol. 39, No. 4, 2013, str. 552-569.

[78] Mende, T., Koschke, R., "Effort-aware defect prediction models.", in CSMR, Capilla, R., Ferenc, R., Dueñas, J. C., (ur.), 2010, str. 107-116.

[79] Zhou, Y., Xu, B., Leung, H., "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems", Journal of Systems and Software, Vol. 83, No. 4, 2010, str. 660-674.

[80] Zhang, H., "An investigation of the relationships between lines of code and defects.", in ICSM. IEEE, 2009, str. 274-283.

[81] Dejaeger, K., Verbraken, T., Baesens, B., "Toward comprehensible software fault prediction models using bayesian network classifiers.", IEEE Trans. Software Eng., Vol. 39, No. 2, 2013, str. 237-257.

[82] Zhang, H., Cheung, S. C., "A cost-effectiveness criterion for applying software defect prediction models", in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, str. 643–646.

[83] Christen, P., Goiser, K., "Quality and complexity measures for data linkage and deduplication.", in Quality Measures in Data Mining, ser. Studies in Computational Intelligence, Guillet, F., Hamilton, H. J., (ur.). Springer, 2007, Vol. 43, str. 127-151.

[84] Gyimothy, T., Ferenc, R., Siket, I., "Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE Trans. Softw. Eng., Vol. 31, No. 10, Oct. 2005, str. 897–910.

[85] Bachmann, A., Bird, C., Rahman, F., Devanbu, P., Bernstein, A., "The missing links: Bugs and bug-fix commits", in Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE '10. New York, NY, USA: ACM, 2010, str. 97–106.

[86] Bird, C., Bachmann, A., Aune, E., Duffy, J., Bernstein, A., Filkov, V., Devanbu, P., "Fair and balanced?: Bias in bug-fix datasets", in Proceedings of ESEC/FSE '09. New York, NY, USA: ACM, 2009, str. 121–130.

[87] Baeza-Yates, R. A., Ribeiro-Neto, B., Modern Information Retrieval. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[88] Fischer, M., Pinzger, M., Gall, H., "Analyzing and relating bug report data for feature tracking", in Proceedings of the 10th Working Conference on Reverse Engineering, ser. WCRE '03. Washington, DC, USA: IEEE Computer Society, 2003, str. 90–.

[89] D'Ambros, M., Lanza, M., Robbes, R., "An extensive comparison of bug prediction approaches", in MSR, 2010, str. 31-41.

[90] Nguyen, A. T., Nguyen, T. T., Nguyen, H. A., Nguyen, T. N., "Multi-layered approach for recovering links between bug reports and fixes", in Proceedings of FSE '12, 2012, str. 63:1–63:11.

[91] Sureka, A., Lal, S., Agarwal, L., "Applying fellegi-sunter (fs) model for traceability link recovery between bug databases and version archives", in Proceedings of APSEC '11. Washington, DC, USA: IEEE Computer Society, 2011, str. 146–153.

[92] Wu, R., Zhang, H., Kim, S., Cheung, S.-C., "Relink: Recovering links between bugs and changes", in Proceedings of ESEC/FSE '11. New York, NY, USA: ACM, 2011, str. 15–25.

[93] Bissyande, T. F., Thung, F., Wang, S., Lo, D., Jiang, L., Reveillere, L., "Empirical evaluation of bug linking", in Proceedings of CSMR '13. Washington, DC, USA: IEEE Computer Society, 2013, str. 89–98.

[94] Song, Q., Jia, Z., Shepperd, M., Ying, S., Liu, J., "A general software defect-proneness prediction framework", IEEE Trans. Softw. Eng., Vol. 37, May 2011, str. 356–370.

[95] Alpaydin, E., Introduction to Machine Learning, 2nd ed. The MIT Press, 2010.

[96] Weiss, G. M., "Mining with rarity: A unifying framework", SIGKDD Explor. Newsl., Vol. 6, No. 1, Jun. 2004, str. 7–19.

[97] Provost, F., "Machine learning from imbalanced data sets 101 (extended abstract)", 2000.

[98] He, H., Garcia, E. A., "Learning from imbalanced data", IEEE Trans. on Knowl. and Data Eng., Vol. 21, No. 9, Sep. 2009, str. 1263–1284.

[99] Chawla, N. V., “Data mining for imbalanced datasets: An overview”, 2005.

[100] Jiang, Y., Cukic, B., Ma, Y., “Techniques for evaluating fault prediction models”, Empirical Softw. Engg., Vol. 13, October 2008, str. 561–595.

[101] Chen, C., Breiman, L., “Using random forest to learn imbalanced data”, 2004.

[102] Bhowan, U., Johnston, M., Zhang, M., Yao, X., “Evolving diverse ensembles using genetic programming for classification with unbalanced data.”, IEEE Trans. Evolutionary Computation, Vol. 17, No. 3, 2013, str. 368-386.

[103] Rubinić, E., Mauša, G., Galinac Grbac, T., “Software defect classification with a variant of nsga-ii and simple voting strategies”, in Search-Based Software Engineering, ser. Lecture Notes in Computer Science, Barros, M., Labiche, Y., (ur.). Springer International Publishing, 2015, Vol. 9275, str. 347-353.

[104] Van Hulse, J., Khoshgoftaar, T. M., Napolitano, A., “Experimental perspectives on learning from imbalanced data”, in Proceedings of the 24th International Conference on Machine Learning, ser. ICML ’07. New York, NY, USA: ACM, 2007, str. 935–942.

[105] Kamei, Y., Monden, A., Matsumoto, S., Kakimoto, T., ichi Matsumoto, K., “The effects of over and under sampling on fault-prone module detection”, 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Vol. 0, 2007, str. 196-204.

[106] Giger, E., Pinzger, M., Gall, H. C., “Comparing fine-grained source code changes and code churn for bug prediction”, in Proceedings of the 8th Working Conference on Mining Software Repositories, ser. MSR ’11. New York, NY, USA: ACM, 2011, str. 83–92.

[107] Giger, E., Pinzger, M., Gall, H., “Using the gini coefficient for bug prediction in eclipse”, in Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution, ser. IWPSE-EVOL ’11. New York, NY, USA: ACM, 2011, str. 51–55.

[108] Ohlsson, N., Zhao, M., Helander, M., “Application of multivariate analysis for software fault prediction”, Software Quality Control, Vol. 7, May 1998, str. 51–66.

[109] Andrews, A., Stringfellow, C., “Quantitative analysis of development defects to guide testing: A case study”, Software Quality Control, Vol. 9, 2001, str. 195–214.

[110] Khoshgoftaar, T. M., Allen, E. B., Halstead, R., Trio, G. P., “Detection of fault-prone software modules during a spiral life cycle”, in Proceedings of the 1996 International Conference on Software Maintenance, ser. ICSM ’96. Washington, DC, USA: IEEE Computer Society, 1996, str. 69–76.

[111] Hochman, R., Khoshgoftaar, T. M., Allen, E. B., Hudepohl, J. P., “Evolutionary neural networks: A robust approach to software reliability problems”, in Proc. of SRE’97, 1997, str. 13–26.

[112] Krishnan, S., Strasburg, C., Lutz, R. R., Goševa-Popstojanova, K., “Are change metrics good predictors for an evolving software product line?”, in Proceedings of the 7th International Conference on Predictive Models in Software Engineering, ser. Promise ’11. New York, NY, USA: ACM, 2011, str. 7:1–7:10.

[113] Khoshgoftaar, T. M., Seliya, N., “Comparative assessment of software quality classification techniques: An empirical case study.”, Empirical Software Engineering, Vol. 9, No. 3, 2004, str. 229-257.

[114] Zimmermann, T., Nagappan, N., “Predicting defects using network analysis on dependency graphs”, in Proceedings of the 30th International Conference on Software Engineering, ser. ICSE ’08. New York, NY, USA: ACM, 2008, str. 531–540.

[115] Briand, L. C., Daly, J. W., Porter, V., Wüst, J., “A comprehensive empirical validation of product measures for object-oriented systems”, 1998.

[116] Zhang, H., Zhang, X., “Comments on "data mining static code attributes to learn defect predictors".”, IEEE Trans. Software Eng., Vol. 33, No. 9, 2007, str. 635-637.

[117] Menzies, T., Dekhtyar, A., Stefano, J. S. D., Greenwald, J., “Problems with precision: A response to "comments on ’data mining static code attributes to learn defect predictors’"”, IEEE Trans. Software Eng., Vol. 33, No. 9, 2007, str. 637–640.

[118] Shihab, E., Mockus, A., Kamei, Y., Adams, B., Hassan, A. E., “High-impact defects: A study of breakage and surprise defects”, in Proceedings of the 19th ACM SIGSOFT

Symposium and the 13th European Conference on Foundations of Software Engineering, ser. ESEC/FSE '11. New York, NY, USA: ACM, 2011, str. 300–310.

[119] Kaur, I., Kaur, A., "Empirical study of software quality estimation", in Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology, ser. CCSEIT '12. New York, NY, USA: ACM, 2012, str. 694–700.

[120] Banthia, D., Gupta, A., "Investigating fault prediction capabilities of five prediction models for software quality", in Proceedings of the 27th Annual ACM Symposium on Applied Computing, ser. SAC '12. New York, NY, USA: ACM, 2012, str. 1259–1261.

[121] Saxena, P., Saini, M., "Empirical studies to predict fault proneness: A review", International Journal of Computer Applications, Vol. 22, No. 8, May 2011, str. 41–45, published by Foundation of Computer Science.

[122] Liu, Y., An, A., Huang, X., Advances in Knowledge Discovery and Data Mining: 10th Pacific-Asia Conference, PAKDD 2006, Singapore, April 9-12, 2006. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Boosting Prediction Accuracy on Imbalanced Datasets with SVM Ensembles, str. 107–118.

[123] Breiman, L., "Random forests", Mach. Learn., Vol. 45, No. 1, Oct. 2001, str. 5–32.

[124] Khoshgoftaar, T. M., Golawala, M., Hulse, J. V., "An empirical study of learning from imbalanced data using random forest.", in ICTAI (2). IEEE Computer Society, 2007, str. 310-317.

[125] Rodriguez, J. J., Kuncheva, L. I., Alonso, C. J., "Rotation forest: A new classifier ensemble method", IEEE Trans. Pattern Anal. Mach. Intell., Vol. 28, No. 10, Oct. 2006, str. 1619–1630.

[126] Amasyali, M. F., Ersoy, O. K., "Classifier ensembles with the extended space forest", IEEE Transactions on Knowledge and Data Engineering, Vol. 26, No. 3, March 2014, str. 549-562.

[127] Palivela, H., Yogish, H., Vijaykumar, S., Patil, K., "A study of mining algorithms for finding accurate results and marking irregularities in software fault prediction", in International Conferenc on Information Communication and Embedded Systems (ICICES), 2013, str. 524–530.

[128] Japkowicz, N., Shah, M., (ur.), Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, 2011.

[129] Harman, M., Islam, S., Jia, Y., Minku, L., Sarro, F., Srivisut, K., "Less is more: Temporal fault predictive performance over multiple hadoop releases", in Proc. of SSBSE'14, 2014, str. 240-246.

[130] Bhowan, U., Johnston, M., Zhang, M., Yao, X., "Reusing genetic programming for ensemble selection in classification of unbalanced data", IEEE Transactions on Evolutionary Computation, 2013.

[131] Elazmeh, W., Japkowicz, N., Matwin, S., Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Evaluating Misclassifications in Imbalanced Data, str. 126–137.

[132] Hastie, T., Tibshirani, R., Friedman, J., The elements of statistical learning: data mining, inference and prediction, 2nd ed. Springer, 2009.

[133] Hamadicharef, B., Guan, C., Ifeachor, E. C., Hudson, N., Wimalaratna, S., "Performance Evaluation and Fusion of Methods for Early Detection of Alzheimer Disease", in International Conference on BioMedical Engineering and Informatics (BMEI2008), Sanya, China, May 2008.

[134] Tabachnick, B. G., Fidell, L. S., Using Multivariate Statistics (5th Edition). Needham Heights, MA, USA: Allyn & Bacon, Inc., 2006.

[135] Han, J., Kamber, M., Pei, J., Data Mining: Concepts and Techniques, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[136] Vapnik, V., Chervonenkis, A., Theory of Pattern Recognition [in Russian]. Moscow: Nauka, 1974, (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

[137] Sayyad Shirabad, J., Menzies, T., "The PROMISE Repository of Software Engineering Databases.", School of Information Technology and Engineering, University of Ottawa, Canada, dostupno na: http://promise.site.uottawa.ca/SERepository 2005.

[138] Khoshgoftaar, T. M., Yuan, X., Allen, E. B., Jones, W. D., Hudepohl, J. P., "Uncertain classification of fault-prone software modules", Empirical Software Engineering, Vol. 7, No. 4, 2002, str. 295-295.

[139] Arisholm, E., Briand, L. C., Johannessen, E. B., "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models", J. Syst. Softw., Vol. 83, No. 1, 2010, str. 2–17.

[140] Shatnawi, R., 0014, W. L., "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process.", Journal of Systems and Software, Vol. 81, No. 11, 2008, str. 1868-1882.

[141] Radjenović, D., Heričko, M., Torkar, R., Živković, A., "Software fault prediction metrics: A systematic literature review", Information and Software Technology, Vol. 55, No. 8, 2013, str. 1397 - 1418.

[142] Li, P. L., Herbsleb, J., Shaw, M., Robinson, B., "Experiences and results from initiating field defect prediction and product test prioritization efforts at abb inc.", in Proceedings of the 28th International Conference on Software Engineering, ser. ICSE '06, 2006, str. 413–422.

[143] Ostrand, T. J., Weyuker, E. J., Bell, R. M., "Where the bugs are", SIGSOFT Softw. Eng. Notes, Vol. 29, No. 4, 2004, str. 86–96.

[144] Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y., Bener, A., "Defect prediction from static code features: current results, limitations, new approaches", Automated Software Engg., Vol. 17, December 2010, str. 375–407.

[145] Nagappan, N., Ball, T., Zeller, A., "Mining metrics to predict component failures", in Proceedings of the 28th International Conference on Software Engineering, ser. ICSE '06, 2006, str. 452–461.

[146] Concas, G., Marchesi, M., Murgia, A., Tonelli, R., "An empirical study of social networks metrics in object-oriented software", Adv. Soft. Eng., Vol. 2010, 2010, str. 4:1–4:21.

[147] Eaddy, M., Zimmermann, T., Sherwood, K. D., Garg, V., Murphy, G. C., Nagappan, N., Aho, A. V., "Do crosscutting concerns cause defects?", IEEE Trans. Softw. Eng., Vol. 34, No. 4, 2008, str. 497–515.

[148] Devine, T. R., Goseva-Popstojanova, K., Krishnan, S., Lutz, R. R., Li, J. J., "An empirical study of pre-release software faults in an industrial product line.", in ICST, 2012.

[149] Steff, M., Russo, B., "Characterizing the roles of classes and their fault-proneness through change metrics.", in ESEM, Runeson, P., Höst, M., Mendes, E., Andrews, A. A., Harrison, R., (ur.). ACM, 2012, str. 59-68.

[150] Caglayan, B., Bener, A., Koch, S., "Merits of using repository metrics in defect prediction for open source projects", in Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, ser. FLOSS '09. Washington, DC, USA: IEEE Computer Society, 2009, str. 31–36.

[151] Bell, R. M., Ostrand, T. J., Weyuker, E. J., "Looking for bugs in all the right places", in Proceedings of the 2006 International Symposium on Software Testing and Analysis, ser. ISSTA '06. New York, NY, USA: ACM, 2006, str. 61–72.

[152] Shivaji, S., Whitehead, J., Ram, A., Kim, S., "Reducing features to improve bug prediction", in Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, ser. ASE '09. Washington, DC, USA: IEEE Computer Society, 2009, str. 600–604.

[153] Knab, P., Pinzger, M., Bernstein, A., "Predicting defect densities in source code files with decision tree learners", in Proceedings of the 2006 International Workshop on Mining Software Repositories, ser. MSR '06, 2006, str. 119–125.

[154] Park, J., Kim, M., Ray, B., Bae, D.-H., "An empirical study of supplementary bug fixes.", in MSR, Lanza, M., Pent, M. D., Xi, T., (ur.). IEEE, 2012, str. 40-49.

[155] Steff, M., Russo, B., "Co-evolution of logical couplings and commits for defect estimation.", in MSR. IEEE, 2012, str. 213-216.

[156] Taba, S. E. S., Khomh, F., Zou, Y., Hassan, A. E., Nagappan, M., "Predicting bugs using antipatterns.", in ICSM. IEEE, 2013, str. 270-279.

[157] Vidacs, L., Beszedes, A., Tengeri, D., Siket, I., Gyimothy, T., “Test suite reduction for fault detection and localization: A combined approach”, in Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on, 2014, str. 204-213.

[158] Oyetoyan, T. D., Cruzes, D. S., Conradi, R., “Transition and defect patterns of components in dependency cycles during software evolution”, in Proceeding of IEEE Conference CSMR-WCRE 2014, Demeyer, S., Binkley, D., Ricca, F., (ur.). IEEE, 2014, str. 283-292.

[159] Rahman, F., Khatri, S., Barr, E. T., Devanbu, P., “Comparing static bug finders and statistical prediction”, in Proceedings of the 36th International Conference on Software Engineering, ser. ICSE 2014. New York, NY, USA: ACM, 2014, str. 424–434.

[160] Kim, S., Zhang, H., Wu, R., Gong, L., “Dealing with noise in defect prediction”, in Proceedings of the 33rd International Conference on Software Engineering, ser. ICSE ’11, 2011, str. 481–490.

[161] Rahman, F., Posnett, D., Devanbu, P., “Recalling the "imprecision" of cross-project defect prediction”, in Proceedings of the ACM SIGSOFT FSE ’12. ACM, 2012, str. 61:1–61:11.

[162] Arisholm, E., Briand, L. C., “Predicting fault-prone components in a java legacy system.”, in ISESE, Travassos, G. H., Maldonado, J. C., Wohlin, C., (ur.), 2006, str. 8-17.

[163] Malhotra, R., Agrawal, A., “Cms tool: Calculating defect and change data from software project repositories”, SIGSOFT Softw. Eng. Notes, Vol. 39, No. 1, 2014, str. 1–5.

[164] Osman, H., Lungu, M., Nierstrasz, O., “Mining frequent bug-fix code changes”, in IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, 2014, str. 343-347.

[165] di Bella, E., Fronza, I., Phaphoom, N., Sillitti, A., Succi, G., Vlasenko, J., “Pair programming and software defects-a large, industrial case study.”, IEEE Trans. Software Eng., Vol. 39, No. 7, 2013, str. 930-953.

[166] Giger, E., D'Ambros, M., Pinzger, M., Gall, H. C., "Method-level bug prediction.", in ESEM, Runeson, P., Höst, M., Mendes, E., Andrews, A. A., Harrison, R., (ur.), 2012, str. 171-180.

[167] Chhillar, R. S., Nisha, "Empirical analysis of object-oriented design metrics for predicting high, medium and low severity faults using mallows cp", SIGSOFT Softw. Eng. Notes, Vol. 36, No. 6, Nov. 2011, str. 1–9.

[168] Denaro, G., Pezze, M., "An empirical evaluation of fault-proneness models", in Proceedings of the Int'l Conf. on Software Engineering, 2002, str. 241-251.

[169] Fukushima, T., Kamei, Y., McIntosh, S., Yamashita, K., Ubayashi, N., "An empirical study of just-in-time defect prediction using cross-project models", in Proceedings of the ACM MSR '14. ACM, 2014, str. 172–181.

[170] Schröter, A., Zimmermann, T., Zeller, A., "Predicting component failures at design time", in Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, ser. ISESE '06. New York, NY, USA: ACM, 2006, str. 18–27.

[171] D'Ambros, M., Lanza, M., Robbes, R., "On the relationship between change coupling and software defects.", in WCRE, Antoniol, G., Pinzger, M., Chikofsky, E. J., (ur.), 2009, str. 135-144.

[172] Catal, C., "Review: Software fault prediction: A literature review and current trends", Expert Syst. Appl., Vol. 38, No. 4, Apr. 2011, str. 4626–4636.

[173] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H., "The weka data mining software: An update", SIGKDD Explor. Newsl., Vol. 11, No. 1, Nov. 2009, str. 10–18.

[174] Pratt, J. W., "Risk Aversion in the Small and in the Large", Econometrica, Vol. 32, No. 1/2, 1964, str. 122–136.

[175] Arrow, K. J., Aspects of the Theory of Risk-Bearing. Helsinki: Yrjö Jahnsson Foundation, 1965.

[176] Howell, D., Fundamental Statistics for the Behavioral Sciences, ser. Business Statistics. Duxbury Press, 1995.

[177] Razali, N., Wah, Y. B., "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests", Journal of Statistical Modeling and Analytics, Vol. 2, No. 1, Jun. 2011.

[178] Kachigan, S., Statistical Analysis: An Interdisciplinary Introduction to Univariate & Multivariate Methods, ser. Análisis multivariable. Radius Press, 1986.

[179] Corder, G., Foreman, D., Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. Wiley, 2009.

[180] Holm, S., "A simple sequentially rejective multiple test procedure", Scandinavian Journal of Statistics, Vol. 6, 1979, str. 65–70.

[181] MATLAB, version 8.3.0 (R2014a). Natick, Massachusetts: The MathWorks Inc., 2014.

[182] Weiß, C. H., "Statsoft, inc., tulsa, ok.: Statistica, version 8", AStA Advances in Statistical Analysis, Vol. 91, No. 3, 2007, str. 339–341.

[183] "IEEE Standard for a Software Quality Metrics Methodology: IEEE std 1061-1998 (R2004)", Tech. Rep., 2004.

[184] Bulmer, M., Principles of Statistics, ser. Dover Books on Mathematics Series. Dover Publications, 1979.

Prilozi

P1 Objektno orijentirane metrike programskog sustava

Prema definiciji norme IEEE 1061-1992, metrika je kvantitativna veličina koja govori u kojoj mjeri neki entitet posjeduje određeno svojstvo [183]. Metrike programskog koda analiziraju programski kod kao entitet, dok svojstva mogu biti raznovrsna, kao što je to opisano u poglavlju 3.1.1. Sve znanstvene discipline se temelje na normiranim kvantitativnim mjerama, a programsko inženjerstvo nastoji postići isto. Brojni stručnjaci i teoretičari rade na definiciji objektivnih i ponovljivih kvantitativnih veličina čija bi uporaba mogla unaprijediti procese poput planiranja budžeta, procjene troškova, testiranja kvalitete, uklanjanja neispravnosti, optimizacije performansi ili dodjele ljudskih resursa. Uspostavna norme izmjere i određivanjem referentnih vrijednosti omogućava usporedbu. Usporediti se može sa uobičajenim veličinama ili sa veličinama u različitim modulima unutar strukture programskog sustava. Temeljem usporedbe se potom može donositi zaključke.

U ovoj disertaciji poseban naglasak je stavljen na metrike proizvoda, odnosno statičke metrike programskog koda. Alatima LOC Metrics i JHawk se prikuplja vrlo širok spektar takvih metrika. Ti alati zadovoljavaju postavljene uvjete koji omogućavaju njihovu implementaciju u automatizirani alat BuCo, kao što je to objašnjeno u poglavlju 3.2.1. Popis metrika koje prikuplja alat LOC Metrics je prikazan u tablici P.1. Riječ je o metrikama čiji izračun je jednotavniji pa se mogu prikupiti iz različitih programskih jezika. Pored broja riječi i linija koda, alat računa i ciklomatičku složenost kao broj linearno nezavisnih putanja kroz programski kod. Ciklomatička složenost je metrika koju je 1976. godine definirao Thomas J. McCabe pa je po njemu i nazvana. Popis metrika koje prikuplja alat JHawk je prikazan u tablici P.2. Riječ je o mnogo širem skupu metrika u odnosu na one koje računa alat LOC Metrics, a neke metrike su svojstvene za objektno orijentirane programske jezike. Pored metrike koje su izražene brojem linija koda, komentara i/ili riječi, postoje i one metrike koje razlikuju metode definirane unutar

ili izvan promatranog razreda, metode koje vraćaju ili ne vraćaju vrijednosti, nasljeđivanje, odnosno nadrazrede i podrazrede, koheziju i uparenost, enkapsulaciju, odnosno rad sa paketima i razinama ovlasti pristupa, polimorfizam, odnosno do koje razine je neki razred prerađen i sl. Tablice P.1 i P.2 sadrže slovne oznake metrika koje predstavljaju akronim na engleskom jeziku te opis temeljem kojeg su ozanke nastale na hrvatskom i engleskom jeziku.

Tablica P.1: Statičke metrike alata LOC Metrics

#	Metrika	Opis
1	LOC	*broj linija koda* (eng. lines of code)
2	SLOC-P	*izvršne linije koda* (eng. physical executable source lines of code)
3	SLOC-L	*logičke linije koda* (eng. logical source lines of code)
4	MVG	*ciklomatička složenost* (eng. McCabe VG complexity)
5	BLOC	*prazne linije* (eng. blank lines of code)
6	C&SLOC	*linije koje sadrže i kod i komentare* (eng. lines with both code and comments)
7	CLOC	*linije koje sadrže samo komentare* (eng. comment lines of code)
8	CWORD	*broj riječi u komentarima* (eng. number of comment words)
9	HCLOC	*broj linija sa komentarima u zaglavlju* (eng. header comments)
10	HCWORD	*broj riječi u komentarima zaglavlja* (eng. header words)

Tablica P.2: Statičke metrike alata JHawk

#	Metrika	Opis
11	AVCC	prosječna ciklomatička složenost svih metoda u razredu (eng. Average Cyclomatic Complexity of all the methods in the class)
12	CCML	*broj linija komentara u razredu* (eng. Total number of comment lines in the class)
13	CCOM	*broj komentara u razredu* (eng. Total Number of Comments in the class)
14	CBO	*uparenost između objekata* (eng. Coupling Between Objects)
15	COH	*kohezija* (eng. Cohesion)
16	DIT	*dubina stabla nasljeđivanja* (eng. Depth of inheritance tree for this class)
17	EXT	*broj poziva vanjskim metodama za razred* (eng. Number of external method calls made from the class)
18	FIN	*ulazna uparenost* (eng. Fan In, Afferent Coupling)
19	FOUT	*izlazna uparenost* (eng. Fan Out, Efferent Coupling)
20	HBUG	*broj Halstead-ovih pogrešaka svih metoda u razredu* (eng. Cumulative Halstead Bugs of all the methods in the class)
21	HEFF	*Halstead-ov napor svih komponenata razreda* (eng. Cumulative Halstead Effort of all the components in the class)
22	HIER	*broj pozvanih metoda koje su definirane u hijerarhiji razreda*

		(eng. Number of called methods that are defined in the hierarchy of the class)
23	HLTH	*Halstead-ova duljina svih komponenata razreda* (eng. Cumulative Halstead Length of all the components in the class)
24	HVOL	*Halstead-ov volumen svih komponenata u razredu* (eng. Cumulative Halstead Volume of all the components in the class)
25	INST	*broj varijabla članica* (eng. Number of instance variables defined in this class)
26	INTR	*broj implementiranih sučelja* (eng. Number of interfaces implemented by this class)
27	MOD	*broj modifikatora* (eng. Number of modifiers)
28	LCOM	*izostanak kohezije među metodama* (eng. Lack of Cohesion of Methods)
29	LCOM2	*izostanak kohezije među metodama 2* (eng. Lack of Cohesion of Methods 2)
30	LMC	*broj poziva lokalnim metodama* (eng. Number of Local method calls)
31	MAXCC	*maksimalna ciklomatička složenost svih metoda u razredu* (eng. Maximum Cyclomatic Complexity of any method in the class)
32	MI	*indeks održivosti* (eng. Maintainability Index)
33	MINC	*indeks održivosti bez komentara* (eng. Maintainability index - no comments)
34	MPC	*uparenost poruka* (eng. Message Passing Coupling)
35	NAME	*ime razreda* (eng. Name of class)
36	NCO	*broj naredbi* (eng. Number of commands)
37	NLOC	*broj linija koda u razredu* (eng. Total number of Lines of Code in the class)
38	NOMT	*broj metoda u razredu* (eng. Number of methods in class)
39	NOS	*broj Java iskaza* (eng. Total Number of Java Statements in class)
40	NQU	*broj upita* (eng. Number of queries)
41	NSUP	*broj nadrazreda uključujući Object* (Number of superclasses including the Object class)
42	NSUB	*broj podrazreda* (eng. Number of subclasses of this class)
43	PACK	*broj uvezenih paketa* (eng. Number of packages imported by this class)
44	R-R	*omjer recikliranja* (eng. Reuse ratio)
45	S-R	*omjer specijalizacije* (eng. Specialization Ratio)
46	RFC	*odziv razreda* (eng. Response for class)
47	SIX	*indeks specijalizacije* (eng. Specialization Index)
48	SUPER	*ime nadrazreda* (eng. Name of Superclass)
49	TCC	*ukupna ciklomatička složenost svih metoda razreda* (eng. Total Cyclomatic Complexity of all the methods in the class)
50	UWCS	*beztežinska veličina razreda* (eng. Unweighted class size)

Način na koji se računaju neke metrika je jasan iz njihova imena, odnosno opisa. Za preostale metrike je preuzet detaljniji opis iz dokumentacije alata JHwak. Uparenost između objekata (CBO) je suma broja razreda koje se referenciraju na promatrani razred i broja razreda koje promatrani razred referencira. Viša vrijednost uparenosti sugerira da će recikliranje promatranog

razreda biti otežana te da je promatrani razred previše ovisan o ostalim razredima. Kohezija (COH) je metrika koja se računa izrazom: $NumRefs/(NOMT \cdot INST)$ pri čemu numRefs je suma broja referenci atributa u svim metodama promatranog razreda, a veće vrijednosti su poželjne. Ulazna uparenost (FIN) je metrika koja se računa kao broj paketa koji se referencira na paket u kojem se nalazi promatrani razred. Izlazna uparenost (FOUT) se računa kao broj paketa na koji se referencira paket razreda koji se analizira. Broj Halstead-ovih pogrešaka svih metoda u razredu (HBUG) je empirijska procjena koja se dobije kao količnik: $HVOL/3000$. Halstead-ov napor (HEFF) je indikator količine vremena koji je potreban programeru kako bi implementirao metodu, a računa se kao umnožak: $HVOL \cdot HDIF$. Halstead-ova težina (HDIF) se pri tome računa kao omjer broja *jedinstvenih operatora* (eng. unique operators, UOR), *broja operanada* (eng. number of operands, NAND) i broja *jedinstvenih operanada* (eng. unique operands UAND) izrazom: $(UOR/2) * (NAND/UAND)$. Halstead-ova duljina i volumen su indikatori veličine metode, a računaju se kao suma operatora i operanada, odnosno kao umnožak: $HLTH \cdot log_2(HVOC)$, pri čemu se Halsead-ov vokabular (HVOC) računa kao suma: $UOR + UAND$. Broj modifikatora (MOD) broji deklaracije koje su *javne* (eng. public), *zaštićene* (eng. protected), *privatne* (eng. private) ili *predefinirane* (eng. default) razine ovlasti unutar promatranog razreda. Izostanak kohezije među metodama (LCOM) se računa izrazom koju je su definirali Henderson i Sellars, a Fraunhofer institut ju navodi pod oznakom LCOM5. Izraz za mjeru LCOM glasi: $(numRefs - NOMT)/INST \cdot (1 - NOMT)$, pri čemu numRefs je suma broja referenci atributa u svim metodama promatranog razreda. Ova inačica metrike izostanka kohezije ima vrijednosti između 0 i 2, niže vrijednosti su bolje, a vrijednost veća od 1 je indikator lošeg koda. Druga metrika izostanka kohezije (LCOM2) se računa kao razlika između broja parova metoda koji dijele referencu na istu varijablu članicu razreda i broja metoda koje to ne dijele.Manje vrijednosti metrike LCOM2, koje su bliže vrijednosti 0, se smatraju boljima. Indeks održivosti (MI) je složena empirijska mjera koja uključuje težinske faktore, logaritme i sinuse metrika HEFF, NOMT, TCC, NOS i CCOM. Pokazalo se da je održavanje razreda: teško za MI manji od 65, umjereno teško za MI između 65 i 85 te lako za MI veći od 85. Uparenost poruka (MPC) se mjeri kao broj vanjskih metoda koje su pozvane unutar svih metoda promatranog razreda. Broj naredbi (NCO) je broj metoda koje ne vraćaju vrijednost, a broj upita (NQU) je broj metoda koje vraćaju neku vrijednost. Omjer recikliranja (R-R) se računa kao omjer broja nadrazreda bez razreda Object i ukupnog broja razreda u stablu nasljeđivanja, ali bez promatranog razreda, odnosno kao količnik: $(NSUP - 1)/(NSUP - 1) + (NSUB + 1)$. Omjer

specijalizacije (S-R) se računa kao omjer broja podrazreda i nadrazreda bez razreda Object, odnosno kao količnik: $NSUB/(NSUP-1)$. Odziv razreda (RFC) se računa kao ukupan broj metoda deklariranih i pozvanih u razredu, odnosno kao zbroj: $NOMT+EXT$. Visoka vrijednost mjere RFC ukazuje na složenost razreda koja može otežati testiranje i održavanje. Indeks specijalizacije (SIX) mjeri do koje razine nasljeđivanja se podrazredi prerađuju. Bez-težinska veličina razreda (UWCS) se računa kao broj varijabla i metoda definiranih unutar razreda, odnosno kao zbroj: $NOMT+INST$. Programski kod se smatra loš ako UWCS ima vrijednost preko 100.

Tablica P.3 sadrži općenite metrike programskog koda koji su sadržani u gotovim skupovima podataka koje generira alat BuCo. Tu se nalaze tri metrike koje služe za identifikaciju programskog modula, odnosno datoteke koja sadrži izvorni kod. Važno je naglasiti da svaka inačica nekog projekta predstavlja jedan skup podataka. Kombinacija metrika *putanja*, *produkt* i *inačica* je stoga jedinstveni identifikator za svaku datoteku koju se analizira i čije metrike se prikuplja alatima LOC Metrics i JHawk. Zavisna varijabla u skupovima podataka je broj neispravnosti za svaku analiziranu datoteku. Broj neispravnosti nastaje nakon što se veze neispravnosti i predaja pretvore u veze neispravnosti i datoteka. Metrika BN_i se dobije tako da se prebroje jedinstvene veze neispravnosti i datoteka. Metrika Status se dobije tako da se BN_i pretvori u binarnu klasu nakon što se odredi granična vrijednost neispravnosti (GVN) koja dijeli te dvije klase.

Tablica P.3: Općenite metrike programskog koda

#	Metrika	Opis
Opis programskog modula		
1	Putanja	jedinstvena identifikacijska oznaka javnog razreda u datoteci
2	Produkt	ime analiziranog projekta otvorenog koda
3	Inačica	oznaka inačice analiziranog projekta otvorenog koda
Zavisna varijabla sklonosti neispravnostima		
4	BN_i	izlazni broj neispravnosti za promatranu datoteku
5	Status	sklonost neispravnostima koja može biti SNP (1) ili NNP (0)

P2 Statistički testovi empirijskog istraživanja

Tablica P.4 prikazuje mjere deskriptivne statistike izračunate za grupe rezultata izraženih mjerom GM s obzirom na udio manjinske klase u skupu podataka (%SNP). Prikazane su srednja vrijednost ($\bar{x}$), standardna devijacija (s), prva kvartila (q_1), medijan (q_2), treća kvartila (q_3), zakrivljenost (γ) i spljoštenost (κ) za svaku metodu strojnog učenja. Iz vrijednosti srednje vrijednosti, medijana te standardne devijacije i interkvartilnog raspona se može potvrditi da razina neujednačenosti utječe na performanse svih metoda strojnog učenja. Najmanja i najveća razina srednje vrijednosti se razlikuje od 0.25 kod logističke regresije i 0.29 kod naivnog Bayesa do 0.44 kod rotirajuće šume i 0.47 kod slučajne šume. Standardna devijacija je dovoljno malena da se vidi značajnost u promjeni rezultata između grupa. Točnije, standardna devijacija je za cijeli jedan red veličine manja u odnosu na srednju vrijednost. Jedina iznimka je grupa rezultata u kojoj je najmanji udio manjinske klase, ispod 3%. Za tu grupu rezultata je već ustanovljeno da je područje nestabilnijih rezultata u poglavlju 6.2, a ovime je dodatno potvrđeno. Zakrivljenost rezultata je u većini slučajeva unutar raspona malene [-0.5, 0.5] i umjerene zakrivljenosti [-1, 1] [184]. Jedine iznimke su naivni Bayes za grupu sa manje od 3% manjinske klase i rotirajuća šuma za grupe sa manje od 12% manjinske klase. Spljoštenost rezultata koja se udaljava od vrijednosti 3 ukazuje na odudaranje od normalne distribucije. Rezultati pretežno imaju koeficijent spljoštenosti manji od 3, a u rijetkim slučajevima je veća od 3. Naivni Bayes ima najveću vrijednost spljoštenosti od čak 13.13 za grupu sa manje od 3% manjinske klase.

Rezultati Kruskal-Wallis testa između rezultata mjere GM grupiranih s obzirom na razinu neujednačenosti su prikazani tablicama P.5, P.6, P.7 i P.8. Test je korišten uz parametar pouzdanosti $\alpha = 0.05$, odnosno razinu pouzdanosti od 95%. Tablice prikazuju broj uzoraka, sumu rangova i očekivani rang u svakoj od grupa rezultata, te ukupni medijan, broj stupnjeva slobode, H vrijednost, p-vrijednost i hi-kvadrat (χ^2) za *opći* (eng. omnibus) test Kruskal-Wallis. Za svaku metodu strojnog učenja je proveden jedan test koji analizira 11 grupa rezultata sa različitim razinama neujednačenosti, koje su evidentirane tablicom 6.2. Testovi su provedeni za svaki projekt iz kojeg su dobiveni podaci, kao što se vidi iz vertikalne podjele tablica. Rezultat p-vrijednosti koja je u svim slučajevima manja od 0.01 pokazuje da je odbačena nul-hipoteza i ustanovljeno je da postoji značajna razlika među grupama rezultata. Drugim riječima, to znači da razina neujednačenosti skupova podataka značajno utječe na performanse metoda strojnog učenja. Ova spoznaja daje značaj provedbi daljnje analize utvrđivanja granične razine neujednačenosti skupova podataka.

Tablica P.4: Mjere deskriptivne statistike za rezultate GM grupirane po %SNP

	logistička regresija							naivan Bayesov klasifikator							slučajna šuma						
%SNP	$\bar{x}$	s	q_1	q_2	q_3	γ	κ	$\bar{x}$	s	q_1	q_2	q_3	γ	κ	$\bar{x}$	s	q_1	q_2	q_3	γ	κ
3%	0.43	0.24	0.33	0.50	0.60	-0.71	2.54	0.78	0.14	0.71	0.78	0.86	-1.94	13.13	0.24	0.22	0.00	0.29	0.43	0.12	1.50
6%	0.54	0.10	0.48	0.55	0.60	0.00	2.98	0.69	0.07	0.65	0.69	0.74	0.03	2.93	0.48	0.10	0.41	0.47	0.55	0.03	2.78
9%	0.54	0.06	0.50	0.54	0.57	-0.21	3.10	0.64	0.06	0.60	0.64	0.68	-0.35	2.77	0.52	0.08	0.46	0.52	0.57	-0.05	2.52
12%	0.48	0.06	0.43	0.48	0.52	-0.16	2.32	0.58	0.05	0.55	0.59	0.62	0.03	2.45	0.49	0.06	0.44	0.49	0.53	-0.08	2.41
15%	0.57	0.05	0.54	0.57	0.60	0.13	2.93	0.59	0.06	0.56	0.60	0.63	-0.24	2.65	0.59	0.04	0.57	0.59	0.62	0.47	3.62
18%	0.57	0.03	0.55	0.57	0.59	0.11	2.58	0.55	0.04	0.52	0.55	0.57	0.13	2.48	0.67	0.04	0.64	0.68	0.70	-0.12	2.32
21%	0.53	0.09	0.43	0.55	0.61	-0.35	1.65	0.56	0.06	0.49	0.58	0.61	-0.34	1.84	0.58	0.07	0.50	0.60	0.64	-0.42	1.83
25%	0.56	0.08	0.48	0.56	0.64	-0.02	1.22	0.54	0.05	0.50	0.53	0.57	0.15	1.88	0.62	0.06	0.56	0.62	0.67	0.02	1.43
30%	0.60	0.03	0.58	0.60	0.61	-0.32	4.28	0.50	0.03	0.48	0.50	0.53	-0.31	2.71	0.73	0.02	0.71	0.73	0.75	-0.35	2.46
40%	0.65	0.04	0.62	0.66	0.67	-0.67	2.32	0.51	0.03	0.49	0.51	0.54	-0.18	3.04	0.71	0.04	0.68	0.72	0.73	-0.89	2.75
50%	0.68	0.01	0.68	0.69	0.69	-0.12	2.83	0.49	0.03	0.47	0.48	0.50	0.18	2.57	0.71	0.01	0.70	0.72	0.72	-0.08	3.36

	rotirajuća šuma							metoda potpornih vektora						
%SNP	$\bar{x}$	s	q_1	q_2	q_3	γ	κ	$\bar{x}$	s	q_1	q_2	q_3	γ	κ
3%	0.30	0.25	0.00	0.35	0.50	7.36	1.71	0.00	0.03	0.00	0.00	0.00	-0.04	56.17
6%	0.51	0.10	0.44	0.51	0.59	2.55	2.79	0.01	0.04	0.00	0.00	0.00	-0.08	7.76
9%	0.55	0.07	0.50	0.56	0.59	1.48	3.34	0.02	0.04	0.00	0.00	0.00	-0.31	3.37
12%	0.49	0.08	0.42	0.50	0.55	-0.05	2.02	0.06	0.06	0.00	0.08	0.11	-0.30	1.41
15%	0.63	0.04	0.60	0.62	0.65	0.54	2.84	0.04	0.05	0.00	0.00	0.10	0.25	1.43
18%	0.64	0.04	0.61	0.64	0.67	-0.36	2.42	0.08	0.05	0.04	0.08	0.12	-0.25	2.14
21%	0.59	0.09	0.50	0.62	0.66	-0.66	1.86	0.09	0.05	0.08	0.09	0.13	-0.54	2.49
25%	0.64	0.07	0.58	0.64	0.70	-0.29	1.38	0.15	0.07	0.09	0.15	0.22	0.01	1.80
30%	0.72	0.02	0.71	0.72	0.74	0.06	2.71	0.24	0.02	0.22	0.24	0.26	-0.15	2.45
40%	0.72	0.03	0.70	0.72	0.74	-0.14	2.53	0.14	0.03	0.12	0.14	0.16	-0.53	2.39
50%	0.74	0.01	0.73	0.74	0.75	-0.10	3.06	0.23	0.02	0.21	0.23	0.24	0.47	2.41

$\bar{x}$ - srednja vrijednost; s - standardna devijacija, q_1 - prva kvartila, q_2 - medijan, q_3 - treća kvartila, γ - zakrivljenost; κ - spljoštenost

Tablica P.5: Rezultati Krukal-Wallis testa za rezultate logističke regresije grupirane po %SNP

Podaci:	**JDT**			**PDE**			**JDT i PDE**		
Raspon %SNP	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang
0% - 3%	2940	7580199	1470	2940	7872414	1470	5880	31043505	2940
3% - 6%	1200	3899272	600	900	2221773	450	2100	12006725	1050
6% - 9%	480	1503478	240	480	1150585	240	960	5290810	480
9% - 12%	360	767930	180	60	135083	30	420	1673693	210
12% - 15%	180	684848	90	240	626155	120	420	2660419	210
15% - 18%	120	440552	60	240	652395	120	360	2292330	180
18% - 21%	180	565848	90	60	242174	30	240	1550554	120
21% - 25%	120	418280	60	60	198388	30	180	1197825	90
25% - 30%	60	256355	30	120	347741	60	180	1265424	90
30% - 40%	240	1198693	120	180	699297	90	420	3731512	210
40% - 50%	60	329315	30	120	436696	60	180	1590676	90
ukupni medijan:	0,533			0,571			0,55		
stupnjevi slobode:	10			10			10		
H:	849,98			247,78			867,55		
p-vrijednost:	< 0.01			< 0.01			< 0.01		
χ^2 :	759,33			409,38			1045,55		

Tablica P.6: Rezultati Krukal-Wallis testa za rezultate naivni Bayes grupirane po %SNP

Podaci:	**JDT**			**PDE**			**JDT i PDE**		
Raspon %SNP	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang
0% - 3%	2940	12008063	1470.495	2940	9586984	1470	5880	42736578	2940
3% - 6%	1200	3482206	600.202	900	2488683	450	2100	11852076	1050
6% - 9%	480	963612	240.081	480	1157164	240	960	4383547	480
9% - 12%	360	442450	180.061	60	86476	30	420	993667	210
12% - 15%	180	247631	90.0303	240	439195	120	420	1472499	210
15% - 18%	120	92511	60.0202	240	274751	120	360	807210	180
18% - 21%	180	174578	90.0303	60	147671	300	240	671513	120
21% - 25%	120	83414	60.0202	60	72541	30	180	330012	90
25% - 30%	60	23641	30.0101	120	95440	60	180	265575	90
30% - 40%	240	111837	120.040	180	161818	90	420	597760	210
40% - 50%	60	14828	30.0101	120	71979	60	180	193035	90
ukupni medijan:	0,696			0,734			0,712		
stupnjevi slobode:	10			10			10		
H:	33729,28			1445,28			4306,02		
p-vrijednost:	< 0.01			< 0.01			< 0.01		
χ^2 :	2528,385			1305,834			3364,23		

Tablica P.7: Rezultati Krukal-Wallis testa za rezultate slučajne šume grupirane po %SNP

Podaci:	**JDT**			**PDE**			**JDT i PDE**		
Raspon %SNP	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang
0% - 3%	2940	5409613	1470	2940	5720725	1478.711	5880	22405285	2940
3% - 6%	1200	4095808	600	900	2377430	452.6667	2100	12728890	1050
6% - 9%	480	1868504	240	480	1682300	241.422	960	7070693	480
9% - 12%	360	1273463	180	60	259593	30.1778	420	3040560	210
12% - 15%	180	869002	90	240	1014168	120.711	420	3776929	210
15% - 18%	120	653072	60	240	935921	120.711	360	3191705	180
18% - 21%	180	829226	90	60	275314	30.1778	240	2167867	120
21% - 25%	120	598369	60	60	269708	30.1778	180	1706404	90
25% - 30%	60	347837	30	120	558218	60.3556	180	1838762	90
30% - 40%	240	1358001	120	180	917552	90.5333	420	4517322	210
40% - 50%	60	341876	30	120	571772	60.3556	180	1859055	90
ukupni medijan:	0,447			0,408			0,432		
stupnjevi slobode:	10			10			10		
H:	3260,88			2317,07			5372,49		
p-vrijednost:	< 0.01			< 0.01			< 0.01		
χ^2 :	2172,29			1900,17			3892,91		

Tablica P.8: Rezultati Krukal-Wallis testa za rezultate rotirajuće šume grupirane po %SNP

Podaci:	**JDT**			**PDE**			**JDT i PDE**		
Raspon %SNP	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang	Broj uzoraka	Suma rangova	Očekivani rang
0% - 3%	2940	5953956	1470	2940	5969148	1470	5880	23910358	2940
3% - 6%	1200	3974339	600	900	2413617	450	2100	12633743	1050
6% - 9%	480	1801731	240	480	1571615	240	960	6727971	480
9% - 12%	360	1102866	180	60	214933	30	420	2611749	210
12% - 15%	180	856034	90	240	956747	120	420	3630216	210
15% - 18%	120	587729	60	240	880889	120	360	2954653	180
18% - 21%	180	765520	90	60	277813	30	240	2058875	120
21% - 25%	120	578165	60	60	272152	30	180	1680444	90
25% - 30%	60	338255	30	120	556363	60	180	1803396	90
30% - 40%	240	1340969	120	180	890605	90	420	4428199	210
40% - 50%	60	345206	30	120	578819	60	180	1863867	90
ukupni medijan:	0,486			0,479			0,484		
stupnjevi slobode:	10			10			10		
H:	2503,17			1878,88			4246,35		
p-vrijednost:	< 0.01			< 0.01			< 0.01		
χ^2 :	1734,62			1525,37			3164,88		

Životopis

Goran Mauša je rođen 1986. godine u Rijeci. Nakon završene gimnazije općeg smjera u Opatiji, upisuje sveučilišni studij elektrotehnike, smjer automatika na Tehničkom fakultetu u Rijeci. Diplomirao je 2010. godine sa izvrsnim prosjekom uz pohvale (magna cum laude). Od prosinca 2010. godine zaposlen je kao znanstveni novak na Tehničkom fakultetu u Rijeci na Zavodu za računarstvo pod vodstvom doc. dr. sc. Tihane Galinac Grbac. U nastavi je involviran kao asistent iz kolegija čiji fokus je programiranje, programsko inženjerstvo, baze podataka i teorija informacija. Doktorski studij računarstva upisuje 2011. godine na Fakultetu elektrotehnike i računarstva u Zagrebu pod mentorstvom prof. dr. sc. Bojane Dalbelo Bašić. Područje njegovog istraživačkog rada je predviđanje programskih neispravnosti uporabom metoda mekog računarstva. Sudjeluje na znanstveno-istraživačkim projektima "Analiza i inovativni pristupi razvoju, upravljanju i primjeni kompleksnih softverskih sustava" te "Evolving Software Systems: Analysis and Innovative Approaches for Smart Management" čiji je voditelj doc. dr. sc. Tihana Galinac Grbac. Član je strukovne asocijacije IEEE, sudjelovao je kao član radionice namijenjenoj doktorandima na konferenciji ESEC/FSE 2011, kao recenzent u konferencijama Eurocon 2012 i Mipro, te kao organizacijski voditelj radionice SQAMIA 2014. Rezultate svoga istraživanja je objavio u 10 radova u zbornicima međunarodnih skupova i jednom radom u međunarodnom časopisu Computer science and information systems.

Popis objavljenih djela

Rad u časopisu

1. Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., “A Systematic Data Collection Procedure for Software Defect Prediction”, Computer Science and Information Systems Journal, Vol. 13, Issue 1, siječanj 2016., str. 173–197.

Radovi na konferencijama

1. Rubinić, E., Mauša, G., Galinac Grbac, T., “Software Defect Classification with a Variant of NSGA-II and Simple Voting Strategies”, Proceedings of the International conference SSBSE, Graduate Student Track, 2015., str. 347-353, Bergamo, Italija.
2. Mauša, G., Bogunović, N., Galinac Grbac, T., Dalbelo Bašić, B., “Rotation Forest in Software Defect Prediction”, Proceedings of 4th Workshop SQAMIA, 2015., str. 35-43, Maribor, Slovenija.
3. Spahić, D., Mauša, G., Kraljević Pavelić, S., Galinac Grbac, T., “Data Storage and Analysis System for Conducting Biotechnological Experiments”, Proceedings of the International conference MIPRO CTI, 2015., str. 528-536, Opatija, Hrvatska.
4. Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., “Data Collection for Software Defect Prediction – an Exploratory Case Study of Open Source Software Projects”, Proceedings of the International conference MIPRO CTI, 2015., str. 513-519, Opatija, Hrvatska.
5. Mauša, Goran, Perković, Paolo, Galinac Grbac, Tihana, Štajduhar, Ivan, “Techniques for Bug–Code Linking”, Proceedings of 3nd Workshop SQAMIA 2014., str. 47-55, Lovran, Hrvatska.
6. Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., “Software defect prediction with bug-code analyzer – a data collection tool demo”, In: Proceedings of the International conference SoftCOM, 2014., str. 425-426, Split, Hrvatska.
7. Galinac Grbac T., Mauša G., Dalbelo Bašič B., “Stability of Software Defect Prediction in Relation to Levels of Data Imbalance”, Proceedings of 2nd Workshop SQAMIA, 2013., str. 1-10, Novi Sad, Srbija.
8. Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., Pavčević, M. O., “Hill Climbing and Simulated Annealing in Large Scale Next Release Problem”, Proceedings of the International conference EuroCon, 2013., str. 452-459, Zagreb, Hrvatska.
9. Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., “Multivariate Logistic Regression Prediction of Fault-proneness in Software Modules”, Proceedings of the International Conference MIPRO, 2012., str. 813-818, Opatija, Hrvatska.
10. Mauša, G., Galinac Grbac, T., Dalbelo Bašić, B., “Overview of Search-based Optimization Algorithms Used in Software Engineering”, Proceedings of the International Conference IN-TECH, 2012., str. 409-412, Rijeka, Hrvatska.

Biography

Goran Mauša was born in Rijeka in 1986. After finishing grammar school in Opatija, he enrolled in the university study programme of electrical engineering, course automation at the Faculty of Engineering, Rijeka. He graduated with excellent grade and praise (magna cum laude) in 2010. He is employed as a junior researcher at the Department of computer engineering, Faculty of Engineering, Rijeka since December 2010 under the mentorship of doc. dr. sc. Tihana Galinac Grbac. As the teaching assistant, he is involved in courses that cover the areas of programming, software engineering, databases and information theory. He started his PhD study of computing at the Faculty of electrical engineering and computing in Zagreb under the mentorship of prof. dr. sc. Bojana Dalbelo Bašić. Software defect prediction using soft computing methods is the main area of his scientific research. He participates in scientific research projects "Analysis and inovative approach to development, management and application of complex software systems" and "Evolving Software Systems: Analysis and Innovative Approaches for Smart Management" led by doc. dr. sc. Tihana Galinac Grbac. He is a member of the IEEE community, participated in PhD workshop at the ESEC/FSE 2011 conference, acted as a reviewer at the Eurocon 2012 and Mipro conferences, and was the organizing chair of SQAMIA 2014 workshop. The results of his research are published in 10 scientific papers in international conferences and one in internationally reviewed journal.