UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Noi modele hibride de Învățare Automată. Aplicații în Ingineria Software

Rezumat

Doctorand: Diana-Lucia Miholca
Conducător de doctorat: Prof. Dr. Gabriela Czibula

Septembrie 2020

Cuprinsul tezei

Cuprinsul rezumatului

Acronime

AGRARM Algoritm Adaptiv de extragere al Regulilor de Asociere Relaționale Graduale. 9, 12, 17, 20–22

AR Regulă de Asociere. 9, 20, 22

ARARM Algoritm Adaptiv de extragere al Regulilor de Asociere Relaționale. 21

AUC Aria de sub curba ROC. 24, 25, 27, 37–39

DynGRAR Algoritm Dinamic de extragere al Regulilor de Asociere Relaționale Graduale. 9, 12, 17, 21, 22, 42

ECS Estimarea Cuplării Software. 8, 10, 12, 13, 28, 30, 33, 34, 41, 42

GRANUM Algoritmul de extragere al Regulilor de Asociere Relaționale Graduale. 9, 10, 12, 17, 19–22, 24

GRAR Regulă de Asociere Relațională Graduală. 9–12, 17–24, 27, 34, 35, 41, 42

IS Inginerie Software. 8–13, 15, 16, 27, 34, 41, 42

LSI Indexarea Semantică Latentă. 11, 30, 32, 35, 37, 38

MLP Perceptron Multistrat. 10–12, 14, 15, 25–27, 35

OAR Regulă de Asociere Ordinală. 9

PCA Analiza Componentelor Principale. 29

PDS Predicția Defectelor Software. 8–13, 15–17, 20–23, 27, 34, 35, 38–42

RAR Regulă de Asociere Relațională. 9, 11, 12, 14, 17–19, 21, 22, 41

RFBN Rețea Neuronală cu Funcții de Bază Radiale. 10, 14, 15

RNA Rețea Neuronală Artificială. 9–15, 23, 24, 27, 28, 34, 35, 39, 41

SOM Rețea cu Auto-Organizare. 14, 15, 29, 31

ÎA Învățare Automată. 8–14, 23, 27, 28, 33, 39, 41, 42

Lista figurilor

Lista publicațiilor

Categoriile menționate sunt în concordanță cu clasificarea efectuată în anul 2014 a jurnalelor[1] și conferințelor[2] în domeniul Informatică.

Publicații științifice internaționale cotate *Web of Science - Science Citation Index Expanded*

1. [75] **Diana-Lucia Miholca**, Gabriela Czibula, Istvan Gergely Czibula. A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. *Information Sciences*, 441, 2018, pp. 152 – 170. (**indexat Web of Science, IF=4.305**)

   Categorie A, 8 puncte.

2. [32] Istvan Gergely Czibula, Gabriela Czibula, **Diana-Lucia Miholca**, Zsuzsanna Onet-Marian. An aggregated coupling measure for the analysis of object-oriented software systems. *Journal of Systems and Software*, 148, 2019, pp. 1 – 20. (**indexat Web of Science, IF=2.278**)

   Categorie A, 4 puncte.

3. [25] Gabriela Czibula, Istvan Gergely Czibula, **Diana-Lucia Miholca**, Liana Maria Crivei. A novel concurrent relational association rule mining approach. *Expert systems with Applications*, 125, 2019, pp. 142 – 156. (**indexat Web of Science, IF=3.768**)

   Categorie A, 4 puncte.

Publicații științifice internaționale cotate *Web of Science, Conference Proceedings Citation Index*

1. [70] **Diana-Lucia Miholca**, Gabriela Czibula, Zsuzsanna Marian and Istvan-Gergely Czibula. An unsupervised learning based conceptual coupling measure. *19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2017, pp. 247 – 254. (**indexat Web of Science**)

   Categorie C, 1 punct.

2. [74] **Diana-Lucia Miholca**, Adrian Onicaș. Detecting depression from fMRI using relational association rules and artificial neural networks. *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2017, pp. 85 – 92. (**indexat Web of Science**)

   Categorie C, 2 puncte.

[1] http://informatica-universitaria.ro/getpfile/16/CSafisat2.pdf;http://hfpop.ro/standarde/doctorat/2014-jurnale.pdf

[2] http://informatica-universitaria.ro/getpfile/16/CORE2013_Exported.xlsx;http://hfpop.ro/standarde/doctorat/2014-conferinte.pdf

3. [69] **Diana-Lucia Miholca**. An improved approach to software defect prediction using a hybrid machine learning model. *20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2018, pp. 443 – 448. (**indexat Web of Science**)

   Categorie C, 2 puncte.

4. [74] **Diana-Lucia Miholca**, Gabriela Czibula, Liana Maria Crivei. A new incremental relational association rules mining approach. *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference (KES)*, Belgrade, Serbia, 2018, pp. 126 – 135. (**indexat Web of Science**)

   Categorie B, 4 puncte.

5. [72] **Diana-Lucia Miholca**, Gabriela Czibula. DynGRAR: A dynamic approach to mining gradual relational association rules. *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23th International Conference (KES)*, Budapest, Hungary, 2019, pp. 10 – 19. (**indexat Web of Science**)

   Categorie B, 4 puncte.

6. [73] **Diana-Lucia Miholca**, Gabriela Czibula. Software defect prediction using a hybrid model based on semantic features learned from the source code. *International Conference on Knowledge Science, Engineering and Management (KSEM)*, Athens, Greece, 2019, pp. 262 – 274. (**indexat Web of Science**)

   Categorie B, 4 puncte.

7. [77] **Diana-Lucia Miholca**, Gabriela Czibula, Vlad Tomescu. COMET: A conceptual coupling based metrics suite for software defect prediction. *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference (KES)*, Verona, Italy, 2020, *acceptat pentru publicare* (**indexat Web of Science**)

   Categorie B, 4 puncte.

Publicații în reviste internaționale

1. [31] Istvan Gergely Czibula, Gabriela Czibula, **Diana-Lucia Miholca**. Enhancing relational association rules with gradualness. *International Journal of Innovative Computing, Information and Control* 13(1), 2017, pp. 289 – 305. (**indexat Scopus**)

   Categorie B, 4 puncte.

2. [29] Istvan Gergely Czibula, Gabriela Czibula, **Diana-Lucia Miholca**, Zsuzsanna Marian. Identifying hidden dependencies in software systems. *Studia Universitatis Babeș-Bolyai Informatica*, 62(1), 2017, pp. 90 – 106. (**indexat Mathematical Reviews**)

   Categorie D, 0.5 puncte.

3. [71] **Diana-Lucia Miholca**. An adaptive gradual relational association rules mining approach. *Studia Universitatis Babes-Bolyai Series Informatica*, 63(1), 2018, pp. 94 – 110. (**indexat Mathematical Reviews**)

   Categorie D, 1 punct.

Punctajul publicațiilor: 42.5 puncte.

Cuvinte cheie

- Învățare Automată
- Modele de clasificare hibride
- Reguli de Asociere Relaționale
- Rețele Neuronale Artificiale
- Predicția Defectelor Software
- Cuplare software
- Cuplare software conceptuală
- Cuplare software agregată
- Metrici software
- Analiza impactului modificărilor software
- Restructurarea software

Introducere

Această teză, intitulată *Noi modele hibride de Învățare Automată. Aplicații în Ingineria Software*, este concentrată pe dezvoltarea unor modele noi de Învățare Automată, unele dintre acestea fiind hibride, și pe aplicabilitatea lor în contextul Inginerie Software (IS), prin rezolvarea unor probleme importante, în particular Predicția Defectelor Software și Estimarea Cuplării Software.

Predicția Defectelor Software (PDS) constă în identificare componentelor software care prezintă defecte. Aceasta are o aplicabilitate largă, asistând măsurarea evoluției proiectelor software, oferind suport administrării proceselor implicate [22], predicția fiabilității produselor software [116], facilitarea testării și ghidarea examinării codului [46]. Toate acestea permit reducerea semnificativă a costurilor implicate în dezvoltarea și mentenanța produselor software [45]. Mai mult, îndeosebi în cazul sistemelor critice, PDS contribuie la detectarea anomaliilor software care ar putea avea un efect negativ asupra vieților umane. În ciuda importanței și a aplicabilității extinse, PDS rămâne o problemă dificilă, îndeosebi în cazul sistemelor complexe, și o foarte activă arie de cercetare [43].

Estimarea Cuplării Software (ECS) constă în constă în evaluarea gradului de dependență dintre componentele software. Există mai multe tipuri de cuplare în literatură. Prevalente dar neexhaustive sunt măsurile care exprimă inter-dependențe *structurale* [6, 83]. Alte tipuri de cuplare sunt cuplarea *conceptuală* [91], cuplare *dinamică*, și cuplarea *logică* [9].

A ști în ce măsură două componente software sunt conectate ajută la o mai bună înțelegere a unui sistem software și, astfel, îi facilitează evoluția și mentenanța. În mod particular, ECS susține analiza impactului modificărilor aduse componentelor software [91, 12], ceea ce este esențial, dat fiind faptul că un proiect este subiectul schimbărilor frecvente. Cuplarea software facilitează și alte activități, precum restructurarea software [30, 70], testarea prin regresie [109] și predicția gradului de mentenabilitate [89].

Atât PDS cât și ECS contribuie la asigurarea calității software, dificultatea acesteia crescând proporțional cu complexitatea a sistemelor software. Defectele software sunt efecte ale unei slabe calități software, în timp ce diferite tipuri de cuplare se află în relație strânsă cu calitatea arhitecturii software. Imperfecțiunile arhitecturale predispun sistemul software la apariția defectelor [33]. În consecință, este foarte probabilă existența unei conexiuni între cuplarea software, care poate indica neajunsuri arhitecturale, și defectele software. Din acest motiv, în această teză, nu am abordat PDS și ECS doar independent, ci și în relație.

Învățare Automată (ÎA) [79] este un subdomeniu al Inteligenței Artificiale (IA). Aceasta are ca scop dezvoltarea unor sisteme care își îmbunătățesc performanța prin experiență astfel încât să permită învățarea, rezonarea și luarea de decizii în afara interacțiunii umane. Astfel de sisteme sunt destinate rezolvării unor probleme complexe. Datorită dimensiunii și complexității în creștere care caracterizează proiectele software, problemele legate de dezvoltarea software sunt, din punct de vedere computațional, dificile. În consecință, soluțiile bazate pe ÎA sunt indicate pentru abordarea unor probleme din domeniul IS, inclusiv a predicției defectelor și a estimării cuplării. Din perspectiva învățării automate, PDS poate fi formulată ca o problemă de clasificare binară constând în discriminarea dintre componentele software prezentând defecte și cele lipsite de defecte. ECS poate, de asemenea, beneficia de ÎA, mai ales pentru evaluarea

cuplării non-structurale [91].

Extragerea regulilor de asociere [93], sau învățarea regulilor de asociere, reprezintă o metodă bazată pe reguli, nesupervizată, de ÎA, pentru identificarea regularităților din date [4] [86] [95]. Extragerea de Regulă de Asociere (AR) clasice [103] constă în descoperirea unor asocieri frecvente ale valorilor atributelor. Așadar, aceasta nu ia în considerare relații potențial semnificative existente între valorile atributelor, cu excepția asocierii lor, a apariției lor împreună [40]. Regulile de Asociere Ordinale (RAO) [15] sunt AR specifice, care exprimă relații de ordine frecvente. Însă alte tipuri de relații informative, diferite de cele de ordine, pot, de asemenea, să existe între valorile atributelor. În aceste condiții, conceptul de Reguli de Asociere Ordinale a fost extinse, definindu-se conceptul de Reguli de Asociere Relaționale (RAR) care surprinde relații generice, inclusiv ordinale, dar nu numai. Comparativ cu regulile de asociere convenționale, cele relaționale sunt mai puternice, mai expresive, permițând o mai bună înțelegere a datelor curente, în vreme ce, tehnici de ÎA pot fi aplicate pentru a învăța pe baza lor cum să prezică date viitoare [105]. Eficiența lor a fost dovedită în domenii variate, incluzând medicina [100], bioinformatica [24] și ingineria software [27, 28]. În domeniul IS, procesor de descoperire al RAR a fost utilizat deja pentru PDS [27] și detecția defectelor de proiectare software [28].

Prima contribuție originală prezentată în acestă teză constă în introducerea conceptului de **extragere a Regulilor De Asociere Relaționale Graduale (GRAR)** [31] în literatura de specialitate. GRAR generalizează RAR prin substituirea relațiilor booleene cu relații graduale. Aceste relații graduale sunt, de fapt, relații fuzzy [54], având asociat gradul în care sunt satisfăcute și moștenind, totodată, toleranța la erori în date. Așadar, extragerea GRAR devine atât mai expresivă, cât și stabilă în raport cu eventualele erori în date, decât extragerea RAR convenționale, fără caracter gradual. Pentru învățarea nesupervizată a GRAR interesate, adică frecvente și suficient de puternice, am propus un algoritm numit GRANUM [31], acesta fiind o adaptare a algoritmului Apriori pentru extragerea AR clasice [5]

Există situații în care seturile de date care se doresc a fi analizate sunt dinamice, prin faptul că mulțimea atributelor poate fi extinsă, în mod incremental, pe măsură ce noi informații devin disponibile. Pentru a actualiza mulțimile de Reguli de Asociere Relaționale Graduale (GRAR) in astfel de cazuri, algoritmul GRANUM ar putea fi reaplicat de fiecare dată când setul de date este extins cu unul sau mai multe noi atribute. Însă reaplicarea acestuia se poate dovedi a fi relativ ineficientă, cu precădere în cazurile în care mulțimea atributelor a fost doar foarte puțin extinsă. Acest fapt ne-a motivat să introducem, ca a doua contribuție originală conceptuală, un algoritm de învățare a GRAR interesante, numit **Algoritm Adaptiv de extragere al Regulilor de Asociere Relaționale Graduale (AGRARM)** [71]. AGRARM este o alternativă semnificativ mai eficientă, din punct de vedere al complexității timp, la reluarea algoritmului Algoritmul de extragere al Regulilor de Asociere Relaționale Graduale (GRANUM) în momentul în care noi atribute devin disponibile. AGRARM adaptează mulțimea GRAR descoperite ca interesante anterior extinderii setului de date, prin includerea unor noi reguli interesante, definite cu ajutorul unor noi atribute.

Seturile de date dinamice pot fi actualizate periodic nu doar prin adăugarea unor noi atribute, ci și prin includerea unor noi instanțe. Pentru ca, în astfel de scenarii, mulțimea GRAR interesante, descoperite anterior, să fie actualizată în mod eficient am propus, ca o a treia contribuție conceptuală, un algoritm numit **Algoritm Dinamic de extragere al Regulilor de Asociere Relaționale Graduale (DynGRAR)** [72]. Astfel, DynGRAR extinde AGRARM, oferind o alternativă eficientă la reluarea GRANUM atunci când setul de date este extins cu noi atribute, noi instanțe sau atât noi atribute, cât și noi instanțe. Domeniile de aplicare care presupun analiza unor seturi de date dinamice sunt numeroase. Un exemplu este domeniul ingineriei software, datorită naturii intrinseci dinamice a procesului de dezvoltare a produselor software.

Cea de-a patra contribuție fundamentală prezentată în această teză constă în **dezvoltarea unor noi modele hibride de ÎA prin combinarea extragerii GRAR cu Rețele Neu-**

ronale Artificiale (RNA). Într-un prim model hibrid de clasificare, numit **HyGRAR** [75], RNA, în particular Perceptroni Multistrat și Rețele Neuronale cu Funcții de Bază Radiale (RBFN), învață, în manieră supervizată, relații binare graduale între atribute. Fiecare dintre aceste relații estimează în ce măsură o combinație de două valori pentru o pereche de atribute plasează o instanță într-una dintre clasele posibile. Pasul ulterior de extragere al GRAR, efectuat de către GRANUM, descoperă GRAR interesante definite cu ajutorul acestor relații graduale. GRAR interesante, de orice lungime, sunt extrase din subseturile de antrenament corespunzând fiecăreia dintre clase. Prin urmare, ele surprind particularitățile fiecăreia dintre clase, formând un fundament pentru efectuarea clasificării efective.

Propunând HyGRAR ca soluție pentru Predicția Defectelor Software (PDS) [75] și dovedind astfel aplicabilitatea practică a modelului hibrid, am adus, de asemenea, contribuții cercetării aplicate în domeniul IS. Propunerea noastră a fost motivată de intuiți că relațiile dintre valorile unor metrici software relevante pot indica gradul de predispoziție al componentelor software de a conține defecte. Așadar, primul pas în aplicarea modelului HyGRAR pentru a prezice defecte software constă în învățarea unor astfel de relații, între valorile metricilor software, prin antrenarea unor RNA pe date etichetate generate de proiecte anterioare ori versiuni anterioare ale unui sistem software aflat în dezvoltare. Relațiile graduale definite între perechi de metrici software sunt, mai apoi, folosite de GRANUM pentru extragerea GRAR interesante care discriminează între componentele prezentând defecte și cele lipsite de defecte, ale sistemului software analizat. În final, în cadrul fazei de clasificare efectivă, o metodă predefinită, euristică, este folosită pentru a clasifica o nouă componentă software ca prezentând sau nu defecte, în funcție de gradele în care aceasta verifică GRAR interesante.

Regula de clasificare euristică, neadaptabilă, ar putea, totuși, reprezenta o limitarea a modelului HyGRAR. Am soluționat aceasta prin **propunerea HyGRAR* ca versiune îmbunătățită a modelului HyGRAR** [69], în care pasul de clasificare este automatizat. îmbunătățirea s-a obținut prin înlocuirea regulii de clasificare neadaptabile cu una adaptabilă, învățată automat de MLP. Așadar, HyGRAR* utilizează MLP nu doar pentru a învăța relații graduale, ci și pentru a învăța cum să clasifice o componentă software pe baza GRAR interesante. În alte cuvinte, MLP efectuează corelații automate între GRAR interesante și clase, eliminând astfel nevoia de a predefini o regulă de clasificare. **HyGRAR*** a fost, de asemenea, **propus și evaluat ca soluție pentru PDS**.

O altă contribuție pe care am adus-o cercetării aplicative constă în abordarea problemei **Estimarea Cuplării Software (ECS)**. O primă contribuție originală în acest sens a fost să propunem **o nouă măsură de cuplare software conceptuală bazată pe ÎA, numită ConC** [70]. ConC exprimă cuplarea conceptuală dintre două componente software ca similaritatea semantică dintre codul sursă corespunzător lor. Informația semantică este extrasă din codul sursă și reprezentată într-un spaţiu vectorial numeric folosind Doc2Vec [56], un model de predicție bazat pe RNA. Măsura de cuplare conceptuală propusă a fost evaluată empiric în contextul **restructurării software** la nivel de pachete.

Deoarece cuplarea conceptuală și cea structurală sunt parțial complementare [9] iar potențialul combinării diferitelor măsuri de cuplare a fost confirmat în literatură [48], am integrat măsura de cuplare conceptuală propusă în **o nouă măsură de cuplare agregată bazată pe ÎA** [32], care include, în același timp, și o măsură de cuplare structurală. Măsura de cuplare agregată este numită ACE și este definită ca o combinație liniară a cuplării conceptuale și a celei structurale. Cuplerea conceptuală este în continuare calculată pe baza reprezentărilor vectoriale învățate Doc2Vec pe baza codului sursă, iar pentru a exprima dimensiunea structurală a cuplării am folosit o adaptare a unei măsuri de cuplare generice din literatură [101]. În acestă teză, ACE a fost evaluată în termenii eficienței ei pentru **analiza impactului modificărilor software**.

Majoritatea defectelor software sunt cauzate de încălcări ale dependențelor implicate în procesele de dezvoltare software. Aceste dependențe pot fi organizaționale sau tehnice, cele din

urmă incluzând cuplarea software [18]. Chiar dacă cuplarea a fost luată în calcul în literatura aferentă problemei PDS, doar dimensiunea ei structurală a fost intens valorificată. Metricile de cuplare structurale au fost, așadar, raportate de numeroase studii ca fiind relevante pentru a prezice defecte software [92]. Totuși, cuplarea logică s-a confirmat experimental ca având un impact superior asupra PDS, comparativ cu cea structurală [18]. Alte dimensiuni ale cuplării, incluzând-o pe cea conceptuală,au fost doar în foarte mică măsură investigate în relație cu defectele software [108]. În aceste condiții, am propus **o abordare hibridă, numită DePSem, pentru predicția defectelor pe baza unor atribute conceptuale, sau semantice, extrase din codul sursă** [73]. Această abordare a deschis studiul nostru în legătură cu interacțiunea dintre cuplarea conceptuală și defectele software, urmând deopotrivă cele două direcții de cercetare prevalente pe tema PDS, și anume dezvoltarea unor tehnici de clasificare îmbunătățite și propunerea unor atribute software relevante.

Din punct de vedere fundamental, DePSem este o adaptare a modelului HyGRAR*. Acesta diferă de HyGRAR* prin faptul că relațiile graduale considerate în procesul de extragere al GRAR interesante sunt predefinite și nu învățate folosind RNA, însă MLP sunt în continuare utilizați pentru a învăța cum să clasifice o entitate pe baza GRAR interesante. Pentru a prezice defectele software, DePSem înlocuiește metricile convenționale utilizate de HyGRAR și HyGRAR* cu atribute semantice extrase din codul sursă. Metricile clasice includ măsuri de cuplare structurală, pe când atributele semantice contribuie la măsurarea cuplării conceptuale. Pentru a extrage atributele semantice din codul sursă am folosit atât Doc2Vec, cât și Indexarea Semantică Latentă (LSI).

Performanța promițătoare a modelului DePSem ne-a motivat să facem un pas mai departe în direcția valorificării cuplării conceptuale pentru a prezice defecte software. Prin urmare, am propus **o suită de metrici software derivate din cuplarea conceptuală, numită COMET, pentru PDS** [77]. Precum în cazul modelului DePSem, cuplarea conceptuală este calculată pe baza reprezentărilor codului sursă furnizate de către Doc2Vec și LSI. Cele 36 de metrici care compun suita COMET sunt derivate direct din cuplarea conceptuală. Prin urmare, impactul cuplării conceptuale asupra defectelor software este deductibilă din acuratețea clasificării componentelor software reprezentate folosind metricile COMET. Pentru a distinge între relevanța metricilor COMET și performanța algoritmului de clasificare, evaluarea a fost efectuată folosind algoritmi convenționali de ÎA. Aceeași algoritmi de clasificare au fost aplicați și în cazul reprezentării componentelor software cu ajutorul metricilor folosite pe scară largă în literatura specializată pe PDS astfel încât să se deducă relevanța relativă a metricilor COMET pentru predicția defectelor. Prin urmare, prin această ultimă contribuție originală, am urmat, separat, și cea de-a doua direcție de cercetare în legătură cu PDS, și anume propunerea de caracteristici software relevante.

Sumarizând, această teză aduce contribuții conceptuale domeniului ÎA și contribuie, de asemenea, la cercetarea aplicată în domeniul IS. Contribuțiile conceptuale originale sunt: generalizarea tehnicii de extragere a RAR prin propunerea tehnicii de extragere a GRAR [31], dezvoltarea unor algoritmi nesupervizați de extragere a GRAR, în varianta adaptivă și în varianta dinamică [71, 72] și dezvoltarea unor noi modele de clasificare automată prin combinarea extragerii GRAR cu RNA [75, 69, 73]. Contribuțiile aduse cercetării aplicate în domeniul IS sunt: aplicarea noilor modele de ÎA ca soluții pentru PDS [75, 69, 73], propunerea unor abordări bazate pe ÎA pentru estimarea cuplării conceptuale și a celei agregate [70, 32] și definirea unor noi metrici derivate din cuplarea conceptuală ca suport pentru PDS [77].

Contribuțiile originale menționate anterior sunt prezentate în Capitolele 2, 3, 4 și 5 ale acestei teze, ea fiind compusă din cinci capitole al cărei conținut este structurat după cum urmează.

Primul capitol începe prin a prezenta, în prima lui secțiune, o descriere generală a modelelor de ÎA implicate în cercetarea prezentată în această teză, incluzând tehnica de extragere a RAR și diferite tipuri de RNA sau modele bazate pe RNA. Cea dea doua secțiune a primului capitol, Secțiunea 1.2, definește, motivează și discută, în contextul stadiului literaturii de specialitate,

cele două probleme din domeniul IS abordate în această teză: PDS și ECS.

Capitolul 2 prezintă contribuțiile noastre originale la extragerea RAR. Secțiunea 2.1 motivează și definește noul concept de GRAR [31], introduce algoritmul numit GRANUM pentru extragerea GRAR interesante și prezintă o evaluare experimentală a abordării analizei datelor prin extragerea GRAR interesante. Secțiunea 2.2 prezintă versiunea *adaptivă* a algoritmului GRANUM, numită AGRARM, care adaptează mulțimea de GRARinteresante caracterizând seturi de date ale căror mulțimi de atribute sunt extinse incremental [71]. Secțiunea 2.3 descrie versiunea *dinamică* a algoritmului GRANUM, numită DynGRAR, iar Secțiunea 2.4 concluzionează cel de-al doilea capitol și oferă perspective de investigații ulterioare.

Capitolul 3 modelele noastre originale, hibride de ÎAs, ca soluții pentru predicția defectelor software pe baza metricilor software clasice. Așadar, capitolul introduce atât contribuții originale rezultate din hibridizarea GRAR cu RNA, cât și contribuții la cercetarea aplicativă pe tema PDS. Secțiunea 3.1 prezintă primul nostru model hibrid de ÎA, numit HyGRAR, în contextul propunerii lui ca soluție pentru PDS [75]. Secțiunea 3.2 prezintă propunerea noastră pentru o versiune îmbunătățită a modelului HyGRAR [69], în care o regulă de clasificare adaptabilă, învățată de MLP ia locul regulii de clasificare neadaptabile, euristice folosite în versiunea inițială a modelului. HyGRAR* este, și el, propus ca soluție pentru PDS. Concluziile celui de-al treilea capitol sunt formulate în secțiunea lui finală, și anume Secțiunea 3.3. În aceeași secțiune sunt indicate și câteva posibile viitoare direcții de cercetare.

Abordările noastre noi, bazate pe ÎA, pentru ECS sunt introduse în **Capitolul 4**. Abordarea noastră originală constând în exprimarea cuplării conceptuale dintre două componente software pe baza unor reprezentări învățate nesupervizat ale codului sursă aferent lor [70] este propusă în Secțiunea 4.1. Secțiunea 4.2 introduce propunerea originală a unei măsuri de cuplare agregate care combină cuplarea structurală cu cea conceptuală [32]. Secțiunea 4.3 sumarizează cel de-al patrulea capitol și indică posibile direcții de cercetare ulterioară în sfera estimării cuplării software.

Capitolul 5 contribuțiile noastre originale în direcția utilizării cuplării în vederea predicției defectelor software. Secțiunea 5.1 prezintă DePSem [73], o abordare originală pentru problema PDS în care o nouă versiune a modelului nostru de clasificare hibrid, care combină GRAR cu RNA, detectează entitățile software defecte pe baza unor caracteristici conceptuale extrase automat din codul sursă. Noua noastră Our new conceptual coupling based metrics suite proposal for supporting PDS is presented in Section 5.2. The ending section, Section 5.3, draws the conclusions of the last chapter and identifies directions for future investigations.

Finally, the last section of this thesis summarizes and brings together the main areas covered in the thesis. It also lists more general directions for future work.

Capitolul 1

Informații generale

Capitolul curent oferă informații generale despre modelele de Învățare Automată (ÎA) implicate în acestă teză, precum și enunțul, motivația și recenzia literaturii de specialitate pentru cele două probleme din domeniul Inginerie Software (IS) pe care le-am abordat: Predicția Defectelor Software (PDS) și Estimarea Cuplării Software (ECS).

Prima secțiune introduce conceptul de extragere al Regulilor de Asociere Relaționale, pe care l-am extins propunând conceptul mai general de extragere al Regululilor de Asociere Relaționale Graduale, și sumarizează tipurile de Rețele Neuronale Artificiale (RNA) pe care le-am folosit pentru a defini noi măsuri de cuplare, pentru a construi noi modele hibride sau pentru a efectua analize experimentale.

Cea de a doua secțiune formulează, motivează și analizează cele două probleme abordate, PDS și ECS.

1.1 Modele de Învățare Automată folosite

Această secțiune introduce diferite modele de ÎA implicate în cercetarea prezentată în această teză. Prima ei subsecțiune oferă o sumarizare a conceptului de extragere al Regulilor de Asociere Relaționale, pe când cea de a doua discută câteva tipuri de RNA și modele bazate pe RNA.

1.1.1 Reguli de Asociere Relaționale

Extragerea Regulilor de Asociere Relaționale reprezintă o metodă de învățare nesupervizată prin care se descoperă relații generice frecvente care au loc între valorile atributelor unui set de date.

Fie $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ un set de instanțe, fiecare instanță fiind caracterizată de un set de p atribute, $\mathcal{A} = (a_1, \ldots, a_p)$. Fiecărui atribut a_i îi este atribută o valoare dintr-un domeniu nevid D_i, care conține și o valoare nulă, indicându-i inexistența. Notăm cu $\Phi(e_j, a_i)$ valoarea atributului a_i pentru o instanță e_j [66] și cu $\mathcal{R}$ mulțimea tuturor relațiilor care pot fi definite între două domenii D_i și D_j [99].

Definition 1. O Regulă de Asociere Relațională (RAR) [99] este o expresie $(a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}) \Rightarrow (a_{i_1}\ R_1\, a_{i_2}\ R_2\ a_{i_3} \ldots R_{\ell-1}\ a_{i_\ell})$, unde $\{a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}\} \subseteq \mathcal{A} = \{a_1, \ldots, a_p\}$, $a_{i_j} \neq a_{i_k}$, $j, k = 1..\ell$, $j \neq k$ and $R_j \in \mathcal{R}$ este o relație definită pe $D_{i_j} \times D_{i_{j+1}}$, D_{i_j} fiind domeniul atributului a_{i_j}.

Dacă $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ sunt existente pentru k instanțe din $\mathcal{E}$, atunci numim raportul $s = \frac{k}{n}$ *suportul* regulii. Dacă notăm cu $\mathcal{E}' \subseteq \mathcal{E}$ mulțimea tuturor instanțelor pentru care $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ sunt existente, relațiile $\Phi(e_j, a_{i_1}) R_1 \Phi(e_j, a_{i_2})$, $\Phi(e_j, a_{i_2}) R_2 \Phi(e_j, a_{i_3})$, ...,

$\Phi(e_j, a_{i_\ell - 1}) R_{\ell-1} \Phi(e_j, a_{i_\ell})$ având loc pentru fiecare instanță e din $\mathcal{E}'$, atunci numim raportul $c = \frac{|\mathcal{E}'|}{n}$ *confidența* regulii.

Definition 2. Numim o Regulă de Asociere Relațională (RAR) *interesantă* dacă suportul ei, s, și confidența ei, c, sunt mai mari sau egale cu două valori reprezentând praguri minime, s_{min} și c_{min}, adică $s \geq s_{min}$ și $c \geq c_{min}$.

Regulile de Asociere Relaționale interesante exprimă șabloane frecvete în date, fiind, prin urmare, de interes deosebit în analiza datelor. Un algoritm complet și eficient, similar algoritmului Apriori [5], a fost introdus în [99], ca o soluție pentru a extragere toate Regulile de Asociere Relaționale interesante, de orice lungime, dintr-un set de date.

1.1.2 Rețele Neuronale Artificiale

Rețelele Neuronale Artificiale (RNA) [111] sunt modele de ÎA care oferă abordări robuste pentru aproximarea funțiilor [79]. Acestea au fost aplicate cu succes în domenii variate, de la bioarheologie [76] până la neurofiziologie [78].

În cele ce urmează, prezentăm modelele Perceptron Multistrat (MLP) și Rețea Neuronală cu Funcții de Bază Radiale (RFBN), ambele tipuri supervizate de RNA, Rețea cu Auto-Organizare (SOM), un tip nesupervizat de RNA, și Doc2Vec, un model nesupervizat, bazat de MLP și specializat în reprezentarea documentelor text ca vectori numerici.

1.1.2.1 Perceptronul Multistrat

Modelul Perceptron Multistrat (MLP) este un tip de RNA unidirecțională (*feed-forward*). Modelul a fost inspirat de sistemul de învățare biologic, care constă într-o rețea complexă de neuroni interconectați [79]. În mod analog, un MLP este format din multiple straturi interconectate de neuroni, incluzând un strat de intrare, unul de ieșire și unul sau mai multe straturi ascunse. Exceptând neuronii de intrare, fiecare neuron din componența rețelei, numit și unitate computațională, are asociată o *funcție de activare*. Toate conexiunile din rețea sunt ponderate.

Modelul MLP este proiectat pentru a fi antrenat un algoritm de învățare inductivă numit retro-propagare (*backpropagation*) [79]. Învățarea debutează printr-o inițializare aleatoare a ponderilor și constă într-un efort de a asocia exemple de date de intrare cu datele de ieșire corespunzătoare. Pentru aceasta, o eroare este derivată din diferența dintre rezultatele corecte și cele furnizate de rețea, aceasta fiind, mai apoi, propagată înapoi prin rețea. Retro-propagarea erorii este este urmată de calculul gradientului [79], toate ponderile fiind, ulterior, actualizate corespunzător, cu scopul reducerii erorii.

1.1.2.2 Rețele Neuronale cu Funcții de Bază Radiale

Un alt tip supervizat de RNA este modelul RFBN. Acesta a fost introdus în literatură de Broomhead și Lowe [14].

Motivația modelului RFBN se află în răspunsul reglat local al neuronilor biologici, pe când fundamentul său teoretic este dat de interpolarea funcțiilor multivariate [97]. Prin corespondență, rezultatul dat de o RFBN este obținut ca o combinație liniară a unităților de activare ascunse [79], în vreme ce acestea din urmă sunt generate de Funcții de Bază Radiale (FBR).

Există mai mulți algoritmi pentru antrenarea RBFN. Schwenker și co-autorii [97] au clasificat acești algoritmi în 3 scheme de învățare, presupunând una, două sau trei faze. Schema de învățare în două faze este aplicată cel mai frecvent. Aceasta presupune antrenarea separată a stratului ascuns și a ponderilor de ieșire. Centrii funcțiilor radiale sunt, de regulă, învățați în manieră nesupervizată, folosind algoritmi de grupare, iar ponderile sunt învățare în manieră supervizată, prin calculul pseudo-inversei sau prin metoda gradientului descendent.

1.1.2.3 Rețele cu Auto-Oraganizare

O SOM este un tip de RNA nesupervizată, introdus de Kohonen [53]. Modelul este motivat biologic, de funcționarea cortexului vizual.

SOM învață o reprezentare a datelor într-un spațiu cu dimensionalitate redusă. Acestă reprezentare poartă numele de *hartă* Kohonen [102] și păstrează ordinea topologică a instanțelor. Deci, o SOM antrenată indică, deopotrivă, grupări ale instanțelor [55]. În consecință, SOM reunește obiectivele proiectării datelor cu cele ale grupării datelor [49].

Din punct de vedere arhitectural, o SOM constă, spre deosebire de MLP și RBFN, în doar două straturi complet conectate: stratul de intrare și cel ce ieșire, cel de-al doilea modelând harta efectivă și reprezentând starea internă a modelului.

Din perspectiva antrenării, o SOM este antrenată nesupervizat, prin învățare competitivă, în opoziție cu antrenarea prin corecția erorii, care este specifică tipurilor supervizate de RNA.

1.1.2.4 Doc2Vec

Doc2Vec este un model bazat pe un MLP, propus de Le și Mikolov [56]. Acesta permite reprezentarea unei informații textuale de lungime variabilă ca un vector numeric de lungime fixă, fiind, astfel, o alternativă pentru modelele convenționale precum colecție-de-cuvinte (*bag-of-words*) sau colecție-de-n-grame (*bag-of-n-grams*).

Un prim avantaj al modelului Doc2Vec în raport cu modelele tradiționale este faptul că acesta ia în considerare distanța semantică dintre cuvinte [56]. Un avantaj adițional în raport cu reprezentarea de tipul colecție-de-cuvinte (*bag-of-words*) constă în considerarea ordinii cuvintelor, cel puțin în cadrul unor contexte mici.

1.2 Problemele abordate din domeniul Ingineriei Software

Această secțiune prezintă principale probleme din domeniul IS care sunt abordare în această teză, și anume predicția componentelor software prezentând defecte și estimarea cuplării software. În ceea ce urmează, cele două probleme sunt enunțate și motivate, iar stadiul literaturii este rezumat pentru fiecare dintre acestea.

1.2.1 Predicția Defectelor Software

1.2.1.1 Definiția și relevanța problemei

Defectele software sunt erori de logică sau de implementare care cauzează funcționarea în moduri imprevizibile a unui sistem sau producerea de rezultate incorecte de către acesta. Predicția Defectelor Software (PDS) constă în identificarea componentelor software care conțin astfel de defecte.

PDS are o aplicabilitate largă. Ajută administratorii de proiecte în măsurarea modului în care acestea evoluează [22] și oferă suport administrării proceselor implicate, prin evaluarea calității produselor software și a performanței proceselor [22]. De asemenea, contribuie la reducerea costurilor implicate de procesele având ca obiectiv asigurarea calității software [45], de exemplu permițând focusarea pe componentele identificate ca prezentând [46].

În ciuda importanței deosebite și a aplicabilității extinse, predicția automată a defectelor software rămâne o problemă complexă. Una dintre principalele provocări este aceea de a efectua pentru un proiect nou, folosind un model antrenat pe alte proiecte [87]. O altă provocare este aceea de a învăța din seturi de date foarte debalansate.

1.2.1.2 Analiza literaturii

Predicția defectelor în sistemele software este o arie de cercetare foarte activă. De pildă, Hall și co-autorii au considerat, într-un articol sistematic, de tip recenzie a literaturii pe subiectul PDS [43], un număr de 208 studii, toate publicate între anii 2000 și 2010.

Eforturile de cercetare în domeniul PDS urmează una dintre următoarele două direcții: propunerea unor clasificatori performanți [3, 64, 16, 67, 87] sau dezvoltarea unor atribute software relevante, pe baza cărora să se discrimineze între componențe software prezentând defecte și componente software lipsite de defecte [7, 51, 52].

De-a lungul primei direcții, majoritatea studiilor existente [80, 81, 110, 112] au considerat, ca date experimentale, unele dintre seturile de date disponibile în baza de date Promise [96]. Aceasta implică și faptul că numeroase abordări existente sunt bazate pe metricile software Promise [41], care includ, pe lângă altele, măsuri de cuplare structurală. Cuplarea logică a fost folosită și ea, deși într-un număr limitat de de studii [7, 51, 52], pentru predicția defectelor software.

1.2.2 Estimarea Cuplării Software

1.2.2.1 Definiția și relevanța problemei

Cuplarea este o proprietate fundamentală a sistemelor software, între aceasta și calitatea proiectării software existând o corelație puternică. De asemenea, cuplarea are un impact ridicat asupra înțelegerii programelor software. Aceasta exprimă gradul de dependență, fie ea chiar ascunsă, dintre două componente software.

Metricile de cuplare s-au dovedit utile în diferite activități specifice domeniului IS, precum: predicția defectelor software [52], predicția mentenabilității unor arhitecturi orientate spre servicii [89] sau aspecte [68, 23], restructurarea software [30], analiza impactului modificărilor software [91, 12] și testarea prin regresie [109].

1.2.2.2 Analiza literaturii

Conceptul de *cuplare* software a fost introdus în literatură de Stevens și co-autorii [104], în contextul modelării structurate a proiectelor software. Ulterior, cuplarea a fost adaptată programării orientate-obiect (OO). De pildă, un set bine-cunoscut și frecvent folosit de metrici OO, inclusiv metrici exprimând cuplarea structurală, este setul de metrici CK, propus de Chidamber și Kemerer [21]. Un altul este compus din așa-numitele metrici MOOD [36].

Privind tipurile de cuplare exprimate de diferite metrici, cele predominante sunt metricile de cuplare strucurală [6, 83]. Acestea măsoară legăturile de natură structurală dintre componentele software. O analiză unificată a diferitelor metrici existente care exprimă cuplarea structurală a fost propusă de Briand și coautorii [13].

Măsurile de cuplare structurală sunt complementate de măsuri de cuplare *conceptuală*, acestea exprimând gradul de corelație semantică dintre entitățile software [91].

Alte tipuri de cuplare definite în literatura de specialitate sunt cuplarea *dinamică* [8, 61], care ia în considerare conexiunile care apar pe parcursul execuției unui program, și cuplarea *logică* (numită, de asemenea, cuplare evoluționară) [7, 9, 52], aceasta stabilindu-se între entitățile software care sunt modificete împreună, pe parcursul evoluției proiectului software.

Capitolul 2

Contribuții la Extragerea Regulilor de Asociere Relaționale

Acest capitol prezintă contribuțiile originale conceptuale aduse analizei datelor prin extragerea de Reguli de Asociere Relaționale (RAR). Contribuțiile constau în extinderea RAR convenționale și propunerea, astfel, a conceptului de Reguli de Asociere Relaționale Graduale (GRAR) [31], precum și dezvoltarea unei versiuni *adaptive* [71] și a uneia *dinamice* [72] pentru algoritmul de extragere a GRAR.

Prima secțiune, Secțiunea 2.1, motivează și definește noul concept de GRAR [31], introduce algoritmul Algoritmul de extragere al Regulilor de Asociere Relaționale Graduale (GRANUM) pentru extragerea GRAR interesante și prezintă evaluarea experimentală a abordării graduale. Noua abordarea este comparată cu versiunea inițială după criteriile expresivității, abilității de a surprinde reguli relevante din punct de vedere semantic și al robusteții în raport cu eventualele erori prezente în date.

Secțiunea 2.2 introduce și evaluează versiunea *adaptivă* a algoritmului GRANUM, numită Algoritm Adaptiv de extragere al Regulilor de Asociere Relaționale Graduale (AGRARM) [71]. AGRARM descoperă în mod eficient GRAR interesante în seturi de date dinamice ale căror mulțimi de atribute sunt extinse incremental. Această abordare este evaluată în multiple scenarii de extindere a atributelor. Deoarece scopul propunerii a fost oferirea unei alternative eficiente la reaplicarea algoritmului GRANUM în astfel de scenarii, timpul necesar rulării algoritmului AGRARM este comparat cu cel necesar reaplicării algoritmului GRANUM pe setul de date extins.

Secțiunea 2.3 prezintă și evaluează versiunea *dinamică* a algoritmului GRANUM, numită Algoritm Dinamic de extragere al Regulilor de Asociere Relaționale Graduale (DynGRAR) [72]. DynGRAR extinde AGRARM astfel încât să descopere eficient toate GRAR interesante în seturi de date care sunt extinse nu doar cu noi atribute, ci și cu noi instanțe. Secțiunea conține și O evaluare comparativă între DynGRAR și GRANUM, contând în multiple experimente efectuate în diferite variante de extindere și folosind mai multe seturi de date aferente problemei Predicția Defectelor Software (PDS). Prezentarea rezultatelor experimentale este urmată de o comparație cu abordările înrudite existente.

Secțiunea 2.4 concluzionează capitolul și enumeră direcții de cercetare ulterioară.

Abordările prezentare în acest capitol sunt au fost publicate în [31], [71] și [72].

2.1 Extragerea Regulilor de Asociere Relaționale Graduale

Secțiunea curentă prezintă conceptul de extragere de GRAR din date, pe care l-am introdus în [31] ca o extindere a conceptului de extragere de RAR fără de caracter gradual. Scopul propunerii a fost să creăm o abordare mai expresivă și mai stabilă pentru extragerea de RAR.

Folosind relații graduale, mai specific relații fuzzy [54], în locul celor booleene, GRAR țin cont de gradul în care relațiile sunt verificate, moștenind, în același timp, toleranța la erori specifică abordărilor fuzzy.

2.1.1 Motivație

Un incovenient al utilizăriiRAR convenționale în anumite contexte de analiză a datelor este faptul că mulțimea regulilor interesante descoperite este este sensibilă la eventualele erori existente în date [24]. Aceasta deoarece abordarea eșuează în a omite aceste erori. Totuși, erorile sunt frecvente în majoritatea seturilor de date generate de procese din lumea reală [62]. Numeroase studii sugerează că o modelarea convenabilă a datelor afectate de erori [85] este prin utilizarea conceptelor fuzzy [38] . Prin urmare, extinzând RAR prin înlocuirea relațiilor boolene care le compun cu relații fuzzy, ne propunem obținerea unei abordări mai stabile pentru analiza datelor afectate de erori sau imprecizie, adică a datelor aproximate [38].

Mai mult, există situații în care este relevant gradul în care o relație între două atribute este verificată, însă abordarea negradulă îl omite. Relațiile graduale [47] a avantajul funcției de apartenență care exprimă măsura în care o relație este satisfăcută. Așadar, considerând relații fuzzy în procesul de extragere al regulilor, țintim o expresivitate sporită.

Concluzionând, motivația pentru introducerea RAR este bivalentă: este de așteptat ca ele să fie atât mai tolerante la erori, cât și mai expresive.

2.1.2 Model teoretic

Fie $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ un set de instanțe, fiecare instanță fiind caracterizată de un set de p atribute, $\mathcal{A} = (a_1, \ldots, a_p)$. Fiecărui atribut a_i îi este atribuită o valoare dintr-un domeniu nevid D_i, care conține și o valoare nulă, indicându-i inexistența. Notăm cu $\Phi(e_j, a_i)$ valoarea atributului a_i pentru o instanță e_j.

Definition 3. O *relație fuzzy binară* [11] $\mathcal{G}$ între 2 domenii D_i și D_j este definită după cum urmează: $\mathcal{G} = \{< (x, y), \mu_{\mathcal{G}}(x, y) >: x \in D_i, y \in D_j\}$, unde $\mu_{\mathcal{G}} : D_i \times D_j \to [0, 1]$ este o funcție de apartenență care asociază fiecărei perechi $(x, y), x \in D_i, y \in D_j$ *gradul de apartenență* $\mu_{\mathcal{G}}(x, y)$ care indică măsura în care relația $\mathcal{G}$ este satisfăcută.

Notăm cu $\mathcal{F}$ mulțimea tuturos relațiilor fuzzy binare considerate în procesul extragerii de GRAR.

Definition 4. O Regulă de Asociere Relațională Graduală (RARG), $\mathcal{G}Rule$, este o secvență $(a_{i_1}\ \mathcal{G}_1\ a_{i_2}\ \mathcal{G}_2\ a_{i_3} \ldots \mathcal{G}_{\ell-1}\ a_{i_\ell})$, unde $\{a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}\} \subseteq \mathcal{A} = \{a_1, \ldots, a_p\}$, $a_{i_j} \neq a_{i_k}$, $j, k = 1..\ell$, $j \neq k$ și $\mathcal{G}_j \in \mathcal{F}$ este o relație fuzzy definită pe $D_{i_j} \times D_{i_{j+1}}$, D_{i_j} fiind domeniul atributului a_{i_j}.

Gradul de apartenență a unei instanțe $e \in \mathcal{E}$ la o GRAR $\mathcal{G}Rule$ este definit folosind funcția *min* [39] astfel: $\mu_{\mathcal{G}Rule}(e) = \min\{\mu_{\mathcal{G}_j}(\Phi(e, a_{i_j}), \Phi(e, a_{i_{j+1}})), j = 1, 2, \ldots, \ell - 1\}$. Acesta exprimă măsura în care regula este satisfăcută de către instanța e.

a) Dacă $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ nu lipsesc pentru k instanțe din $\mathcal{E}$, atunci numim raportul $s = \frac{k}{n}$ *suportul* regulii $\mathcal{G}Rule$.

b) Dacă $\mathcal{E}' \subseteq \mathcal{E}$ este mulțimea tuturor instanțelor din $\mathcal{E}$ pentru care $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ nu lipsesc și $\mu_{\mathcal{G}Rule}(e) > 0$ pentru fiecare instanță e din $\mathcal{E}'$, atunci numim raportul $c = \frac{|\mathcal{E}'|}{n}$ *confidența* regulii $\mathcal{G}Rule$.

c) Folosind notația de la punctul b), numim raportul $m = \frac{\sum_{e \in E'} \mu_{\mathcal{G}Rule}(e)}{n}$ *gradul de aparteneță al setului de date* $\mathcal{E}$ la regula $\mathcal{G}Rule$.

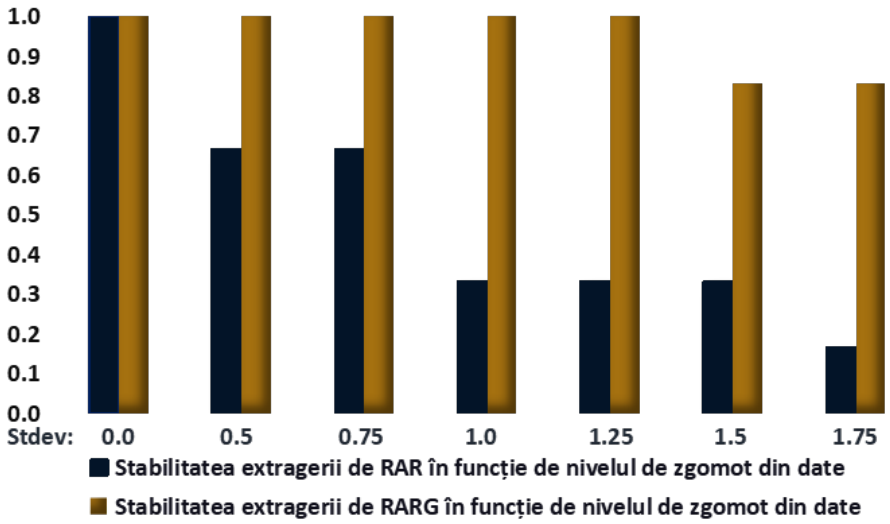


Figure 2.1: Stabilitate la zgomot: RAR versus GRAR

Definition 5. Numim o *GRARro interesantă* în cadrul unui set de date dacă suportul ei s, confidența c și gradul de apartenență m sunt mai mari sau egale decât anumite praguri minime date, $s \geq s_{min}$, $c \geq c_{min}$ și $m \geq m_{min}$.

2.1.3 Algoritmul de extragere a Regulilor de Asociere Relaționale Graduale

Pentru a descoperi toate GRAR interesante într-un set de date, am adaptat algoritmul de învățare al RAR fără caracter gradual introdus în [99], obținând astfel algoritmul GRANUM. Folosind un proces iterativ, GRANUM învață toate GRAR, de orice lungime, în cadrul unui set de date.

GRANUM reduce semnificativ spațiul exponențial de căutare compus din toate GRAR posibile. Acesta ghidează căutarea înspre acele regiuni unde pot fi găsite reguli interesante, evitând restul regiunilor.

Se poate dovedi că algoritmul GRANUM este complet și corect.

2.1.4 Experimente și rezultate

Am evaluat experimental abordarea constând în analiza datelor prin extragerea de GRAR interesante, evidențiind avantajele ei în raport cu abordarea inițială, de extragere de RAR fără caracter gradual.

În primul studiu de caz, am arătat că abordarea graduală poate descoperi reguli care îi sunt inaccesibile abordării convenționale, fie din pricina unor mici erori de aproximare, fie deoarece RAR nu beneficiază de gradul de apartenență care exprim măsura în care regulile sunt verificate.

Un al doilea studiu de caz a evidențiat faptul că abordarea graduală este mai robustă decât cea inițială. Pentru a evalua stabilitatea celor două abordări, am introdus, în manieră incrementală, zgomot Gaussian cu media 0 și deviație standard crescândă în date. Figura 2.1 arată că, după cum era așteptat, stabilitatea scade, pentru ambele abordări, pe măsură ce deviația standard crește, însă mulțimea RAR fără caracter gradual (corespunzându-i culoarea albastru închis) este mai afectată de zgomot.

Pe baza rezultatelor experimentelor efectuate, se poate concluziona că GRAR sunt mai cuprinzătoare, mai expresive și mai puțin și mai puțin sensibile la zgomot decât RAR convenționale.

2.1.5 Comparație cu abordări înrudite

Conceptul de GRAR este unul nou deoarece nu exosta alte abordări care să folosească relații fuzzy [50] în analiza datelor prin extragerea de Reguli de Asociere (AR).

2.2 AGRARM: Un algoritm Adaptiv de extragere a Regulilor de Asociere Relaționale Graduale

În această secțiune prezentăm metoda numiră AGRARM, pe care am introdus-o în [71]. AGRARM descoperă toate GRAR interesante caracterizând un set de date dinamic, al cărui set de atribute este extins prin adăugarea unuia sau mai multor atribute, prin adaptarea mulțimii de reguli descoperite anterior extinderii, astfel încât să se păstreze completitudinea.

2.2.1 Motivație

Algoritmul GRANUM, prezentat pe scurt în Secțiunea 2.1.3, extrage toate GRAR interesante dintr-un set de date descrise de o mulțime dată de atribute. Există, însă, situații în care setul de date este dinamic pe orizontală, în sensul că mulțimea de atribute care caracterizează obiectele componente evoluează în timp. Desigur, în acest caz, pentru a actualiza mulțimea de GRAR interesante, algoritmul de extragere poate fi reaplicat de fiecare dată când mulțimea de atribute se schimbă. Însă aceasta ar putea fi ineficient, mai ales dacă mulțimea de atribute a fost doar foarte puțin extinsă, de pildă prin adăugarea unui singur atribut. Prin urmare, am introdus o alternativă la reaplicarea algoritmului GRANUM în momentul în care setului de date i se adaugă noi atribute, propunând algoritmul AGRARM care adaptează mulțimea de GRAR interesante extrase anterior extinderii astfel încât se obțină mulțimea actualizată a tuturor GRAR care caracterizează setul de date extins.

2.2.2 Metodologie

AGRARM este un algoritm complet care, începând de la mulțimea de GRAR interesante extrase înainte de extinderea setului de date și considerând atributele nou adăugate, actualizează mulțimea regulilor astfel încât aceasta să includă toate GRAR interesante la nivelul setului de date extins.

Așadar, AGRARM începe prin o parcurgere inițială a setului de date extins pentru a identifica noi GRAR binare interesante. Aceste reguli conțin cel puțin un atribut nou adăugat. În iterațiile ulterioare, ele pot fi folosite pentru a forma GRAR de lungimi mai mari.

2.2.3 Experimente și rezultate

Am evaluat comparativ AGRARM și GRANUM reaplicat pe setul de date extins, în termenii eficienței timp. Am considerat trei seturi de date utilizate pentru PDS (Tomcat, Ar și JM1), diferite posibilități de extindere a seturilor de date cu noi atribute și valori multiple pentru pragurile minime de suport, confidență și grad de apartenență.

Rezultatele evaluării experimentale au confirmat că AGRARM extrage toate GRAR interesante din cadrul setului de date extins semnificativ mai rapid decât în cazul reaplicării GRANUM.

2.2.4 Comparație cu abordări înrudite

AGRARM este nou în literatura de specialitate. Abordările existente [82, 35, 84, 19, 113, 60] au în vedere reguli de asociere nerelaționale, iar caracterul lor adaptiv se referă la alte aspecte,

cu excepția Algoritm Adaptiv de extragere al Regulilor de Asociere Relaționale (ARARM) [26], care, însă, extrage RAR fără caracter gradual.

2.3 DynGRAR: Un algoritm dinamic de extragere a Regulilor de Asociere Relaționale Graduale

Prezentăm succint, în secțiunea curentă, algoritmul DynGRAR pe care l-am introdus în [72]. DynGRAR extinde AGRARM astfel încât descoperă GRAR interesante în seturi de date dinamice care sunt extinse incremental atât cu noi atribute, cât și cu noi instanțe.

2.3.1 Motivație

În Secțiunea 2.2 am prezentat algoritmul AGRARM, care descoperă în mod eficient toate GRAR interesante într-un set de date ale cărui mulțime de atribute este extins prin adăugarea unuia sau mai multor noi atribute. Totuși, dinamica setul de date de analizat nu este determinată exclusiv de adăugarea unor noi atribute. Setul de date poate fi extins și prin adăugarea unor noi instanțe.

Prin urmare, am introdus un nou algoritm, numit DynGRAR care să permită extragerea de GRAR în astfel de seturi de date dinamice.

Un posibil domeniu de aplicabilitate pentru DynGRAR este PDS, datorită naturii intrinseci dinamice to a procesului de dezvoltare software.

2.3.2 Metodologie

DynGRAR se compune din două faze succesive.

Prima fază constă în filtrarea regulilor interesante extrase anterior extinderii setului de date, astfel încât să fie păstrate doar regulile care rămân interesante și la nivelul setului de date extins.

Cea de-a doua fază constă în extinderea submulțimii de reguli rezultate din prima fază, cu noi GRAR interesante. Noile GRAR interesante sunt fie GRAR care conțin doar atribute inițiale, dar care nu au fost interesante la nivelul setului de date inițial ci au devenit astfel la nivelul setului de date extins, datorită instanțelor nou adăugare, fie GRAR interesante conținând cel puțin un atribut nou adăugat.

DynGRAR menține completitudinea procedurii de generare de GRAR, fiind de așteptat să fie ca acesta să reprezinte o alternativă mai eficientă din punct de vedere al complexității timp.

2.3.3 Experimente și rezultate

DynGRAR a fost evaluat în comparație cu GRANUM reaplicat pe setul de date extins, în termenii eficienței timp, prin considerarea mai multor seturi de date pentru PDS, a unor posibilități variate de extindere a acestora cu noi atribute și/sau instanțe și a unor valori multiple pentru pragurile minime ale suportului, confidenței și gradului de apartenență.

Fugura 3.2 sumarizează the reduction of the mining time obtained when replacing GRANUM run from scratch with DynGRAR on five different case studies. The mining time has been reduced, in average, with 43% and at most with 93,8%.

2.3.4 Comparație cu abordări înrudite

Algoritmul DynGRAR este nou în literatura de specialitate. Problema extragerii dinamice de GRAR din seturi de date extinse incremental prin adăugarea de noi atribute și noi instanțe

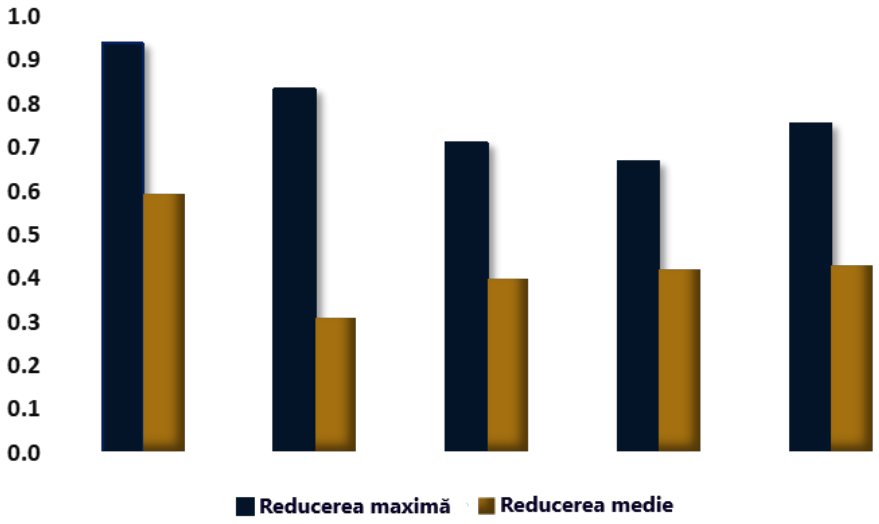


Figure 2.2: Rata de reducere a timpului necesar extragerii GRAR obținută prin aplicarea algoritmului DynGRAR în locul reaplicării algoritmului GRANUM

nu a fost abordată anterior în literatură. Există abordări care presupun extragerea incrementală sau adaptivă de AR clasice [82, 35, 84, 113, 19, 60, 98, 114], însă nu și de GRAR.

2.4 Concluzii și direcții viitoare de cercetare

Această secțiune a sumarizat contribuțiile noastre conceptuale aduse extragerii de RAR din date, și anume propunerea conceptului de extragere de GRAR, care extinde analiza datelor prin extragerea de RAR îmbogățind-o cu o mai mare expresivitate și o mai bună stabilitate în raport cu erorile din date, și dezvoltarea a doi algoritmi de extragere de GRAR, numiți AGRARM și DynGRAR, care învață în manieră dinamică GRAR, în seturi de date care evoluează în timp fiind extinse cu noi atribute și/sau noi instanțe.

Mai multe experimente au fost efectuate cu scopul evaluării performanței relative a celor doi algoritmi adaptivi în raport cu GRANUM reaplicat pe setul de date extins. Rezultatele experimentare reliefează faptul că AGRARM și DynGRAR reduc în mod semnificativ timpul necesar extragerii de GRAR interesante din seturi de date dinamice.

O posibilă direcție de extindere a procesului de extragere de GRAR ar fi includerea relațiilor graduale unare, pe lângă cele binare. Generalizarea ar putea continua, prin adăugarea de relații graduale ternare, cuaternare și așa mai departe.

Investigații ulterioare ar putea fi efectuare și în direcția dezvoltării unei versiuni concurente a algoritmului DynGRAR astfel încât să-i fie sporită eficiența timp.

O altă zonă posibilă pentru cercetare viitoare ar fi folosirea algoritmului DynGRAR în cadrul unei abordări incrementale a problemei PDS.

Capitolul 3

Noi abordări hibride pentru Predicția Defectelor Software

Acest capitol prezintă simultan contribuțiile originale fundamentale aduse domeniului Învățare Automată (ÎA), prin hibridizarea extragerii de Reguli de Asociere Relaționale Graduale (GRAR) cu Rețele Neuronale Artificiale (RNA) în noi modele de clasificare, și cele aduse cercetării aplicate în domeniul aferent problemei Predicția Defectelor Software (PDS), acestea din urmă rezultând din aplicarea modelelor de clasificare hibridă pentru predicția defectelor software, pe baza unor metrici software convenționale.

Secțiunea 3.1 introduce primul nostru model hibrid de ÎA, numit HyGRAR [75], în contextul propunerii acestuia ca predictor al defectelor software. Secțiunea rezumă și o evaluare experimentală a modelului, efectuată considerând 10 seturi de date disponibile public, specifice PDS.

Secțiunea 3.2 introduce HyGRAR* [69], versiunea îmbunătățită a modelului HyGRAR, aceasta fiind, de asemenea, descrisă și evaluată ca soluție pentru PDS. Atât HyGRAR, cât și HyGRAR* combină extragerea de GRAR cu RNA, însă HyGRAR folosește RNA doar anterior pasului de extragere a regulilor, pentru învățarea relațiilor graduale, în timp ce HyGRAR* le folosește și pentru automatizarea pasului de clasificare.

Concluziile capitolului sunt formulate în Secțiunea 3.3, alături de direcții viitoare de cercetare.

Abordările prezentate în acest capitol au fost publicate în [75] și [69].

3.1 HyGRAR: Un model hibrid de predicție al defectelor software pe baza metricilor convenționale

Această secțiune introduce HyGRAR, versiunea inițială a modelului de clasificare hibrid care combină extragerea de GRAR cu RNA, alături de o evaluare experimentală efectuată folosind 10 seturi de date, disponibile public, pentru PDS.

Scurte prezentări ale RNA și GRAR sunt oferite în secțiunile 1.1.2 și 2.1.

3.1.1 Motivație

Enunțul și motivația problemei PDS sunt prezentate în Secțiunea 1.2.1. În propunerea modelului HyGRAR ca soluție pentru PDS, am presupus, intuitiv, faptul că relațiile dintre diferite metrici relevante ar putea fi relevante în a indica vulnerabilitatea față de defecte software. Mai mult, am considerat că a învăța astfel de relații în locul predefinirii lor este de preferat.

3.1.2 Metodologie

Ideea principală pe care se bazează dezvoltarea modelului HyGRAR este următoarea.

Mai întâi, RNA învață din experiența proiectelor anterioare sau a versiunilor anterioare ale unui proiect curent, relații graduale definite între perechi de metrici software care fac componentele software vulnerabile față de apariția defectelor software.

Apoi, algoritmului Algoritmul de extragere al Regulilor de Asociere Relaționale Graduale (GRANUM) i se furnizează aceste relații și, pornind de la acestea, extrage separat GRAR interesante care caracterizează subseturile compuse din componentele software defecte și, respectiv, din cele lipsite de defecte ale setului de date de antrenament corespunzător proiectului curent (sau versiunii curente).

În final, în faza de clasificare, pe baza GRAR interesante și discriminatorii, cu ajutorul unei metode predefinite, o nouă entitate software este clasificată ca fiind defectă sau, dimpotrivă, neafectată.

În concordanță cu cele menționate anterior, pentru a determin dacă o entitate software prezintă sau nu defecte, se efectuează următorii pași:

1. Preprocesarea datelor, implicând selectarea celor mai relevante atribute și balansarea datelor.
2. Construirea clasificatorului HyGRAR prin două faze de antrenare succesive:

   Faza 1. Învățarea relațiilor graduale între fiecare două metrici software, în raport cu vulnerabilitatea față de defecte.

   Faza 2. Extragerea de GRAR interesante care caracterizează fiecare dintre două clase posibile.
3. Clasificarea efectivă (sau testarea).

3.1.3 Experimente și rezultate

3.1.3.1 Seturi de date

În evaluarea experimentală a modelului HyGRAR s-au folosit 10 seturi de date publice. Mai întâi, am considerat 5 seturi de date (Ar1, Ar3, Ar4, Ar5 și Ar6 [96]) corespunzând sistemului Ar, implementat în C, evaluând astfel HyGRAR dintr-o perspectivă inter-versiuni. Am evaluat HyGRAR și din perspectiva inter-proiecte, considerând 3 sisteme dezvoltate în paradigma orientată-obiect: JEdit, Ant și Tomcat. Pentru JEdit, s-a considerat 3 versiuni diferite.

Am selectat un număr relativ mic de metrici software dintre cele disponibile, după criteriul corelației sintre acestea și eticheta de clasă.

3.1.3.2 Evaluare

Evaluarea a fost efectuată folosind metoda de validate încrucișată *leave-one-out* și am calculat mai mulți indicatori de performanță (incluzând acuratețea, precizia, specificitatea, senzitivitatea), însă pentru comparații am folosit predominant Aria de sub curba ROC (AUC), întrucât este considerată ca fiind cea mai bună măsură pentru compararea predictorilor de defecte software [37].

3.1.3.3 Rezultate

Considerând 17 alte abordări existente care raportează rezultate pe seturile de date Ar, prezentate în [1, 10, 88, 17, 107, 3, 112, 81, 65], Figura 3.1 ilustrează valorile pentru AUC

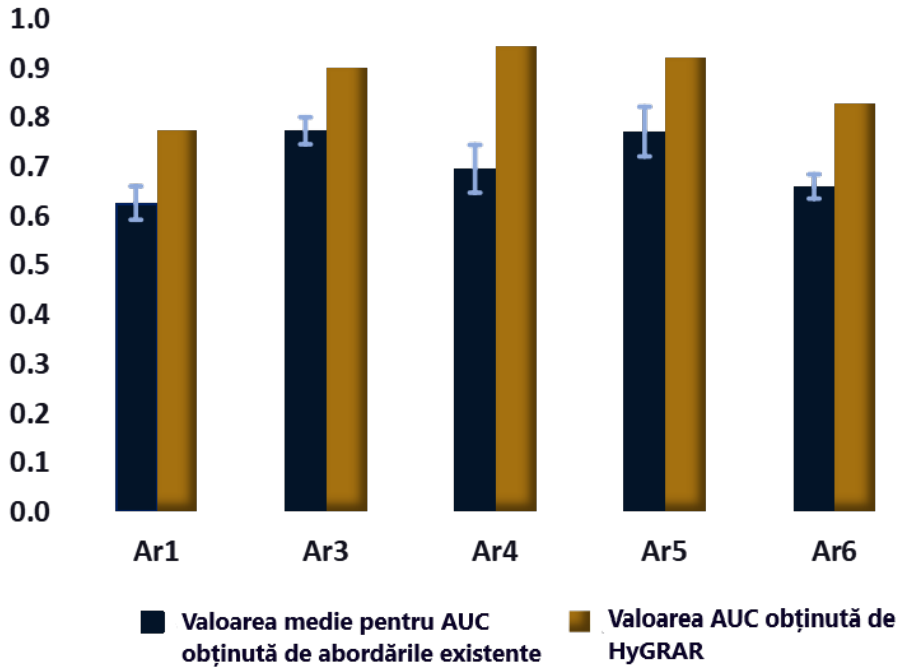


Figure 3.1: Performanța relativă a modelului HyGRAR evaluată pe seturile de date Ar

obținute de HyGRAR în raport cu valorile medii pentru AUC obținute în cazul abordărilor înrudite, pentru care erorile standard sunt, de asemenea, reprezentate. HyGRAR a obținut valori pentru AUC mai mari decât valorile medii pentru acesta obținute de abordările alternative considerate. Modelul nostru depășește toate celelalte abordări pentru 3 versiuni și se plasează pe primele două locuri pentru toate cele 5 versiuni.

Luând în calcul 15 alte abordări pentru efectuarea comparației [67, 64, 16, 80, 87], HyGRAR a obținut valori pentru AUC care au depășit nu doar media, cât și maximul valorilor obținute de abordările existente care raportează rezultate pe aceleași seturi de date.

Rezultatele obținute indică că, dintre cele 10 studii de caz, în cazul a 8 dintre ele, HyGRAR s-a dovedit a fi cel mai performant clasificator, iar pentru celelalte două se se situează în topul celor mai buni 2 clasificatori.

3.2 HyGRAR*: Un model hibrid îmbunătățit de predicție al defectelor software pe baza metricilor convenționale

În secțiunea curentă prezentăm versiunea îmbunătățită a modelului HyGRAR, numită HyGRAR*.

Îmbunătățirea este obținută prin înlocuirea regulii euristice de clasificare din faza de testare a versiunii inițiale cu una adaptivă bazată pe Perceptroni Multistrat. Evaluarea experimentală efectuată pe două seturi de date NASA indică faptul că clasificatorul HyGRAR* surclasează atât HyGRAR, versiunea inițială a modelului hibrid, cât și celelalte abordări existente, evaluate pe aceleași două seturi de date.

3.2.1 Motivație

O limitare a abordării HyGRAR este dată de faptul că faza de clasificare nu este adaptivă.

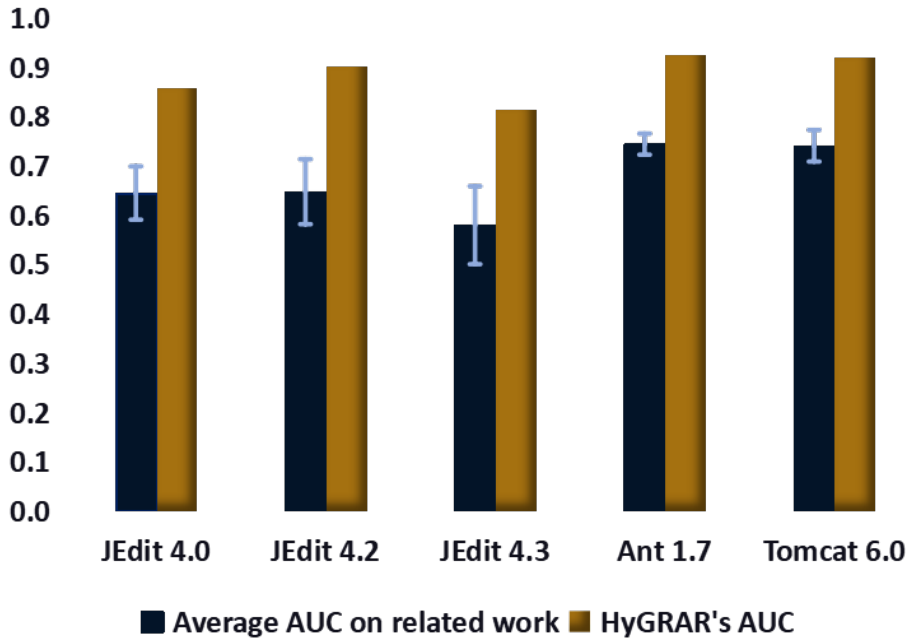


Figure 3.2: Performanța relativă a modelului HyGRAR evaluată pe seturile de date eferente proiectelor software orientate-obiect

Pentru a elimina această limitare, propunem îmbunătățirea fazei de clasificare a modelului HyGRAR prin învățarea automată a regulii de clasificare.

3.2.2 Metodologie

Fazele de preprocesare și cei doi pași de antrenare utilizați în dezvoltarea clasificatorului HyGRAR rămân neschimbate în cazul versiunii HyGRAR*, însă regula de clasificare euristică folosită în faza de clasificare este substituită de una adaptivă an adaptive învățată de MLP.

3.2.3 Experimente și rezultate

3.2.3.1 Seturi de date

Experimentele au fost efectuate pe seturile de date NASA corespunzând proiectului software PC [96], prin considerarea a două studii de caz. Cele două studii de caz corespund seturilor de date PC3 și PC4, acestea fiind cele mai dificile dintre cele patru seturi aferente celor patru versiuni ale proiectului, conform [27].

3.2.3.2 Evaluare

Pentru evaluarea performanței modelului HyGRAR*, s-a folosit aceeași metodologie precum în cazul HyGRAR (a se vedea Secțiunea 3.1.2).

3.2.3.3 Rezultate

Considerând cele două studii de caz, HyGRAR* a surclasat toate cele 9 abordări existente identificate în literatură care raportează rezultate pe aceleași seturi de date [63, 90, 27], precum și versiunea inițială a modelului.

Cele două figuri de mai jos compară HyGRAR* cu abordările existente, după valorile AUC obținute, dat și după acuratețe, dat fiind că majoritatea abordărilor înrudite reportează doar acuratețea ca indicator de performanță.

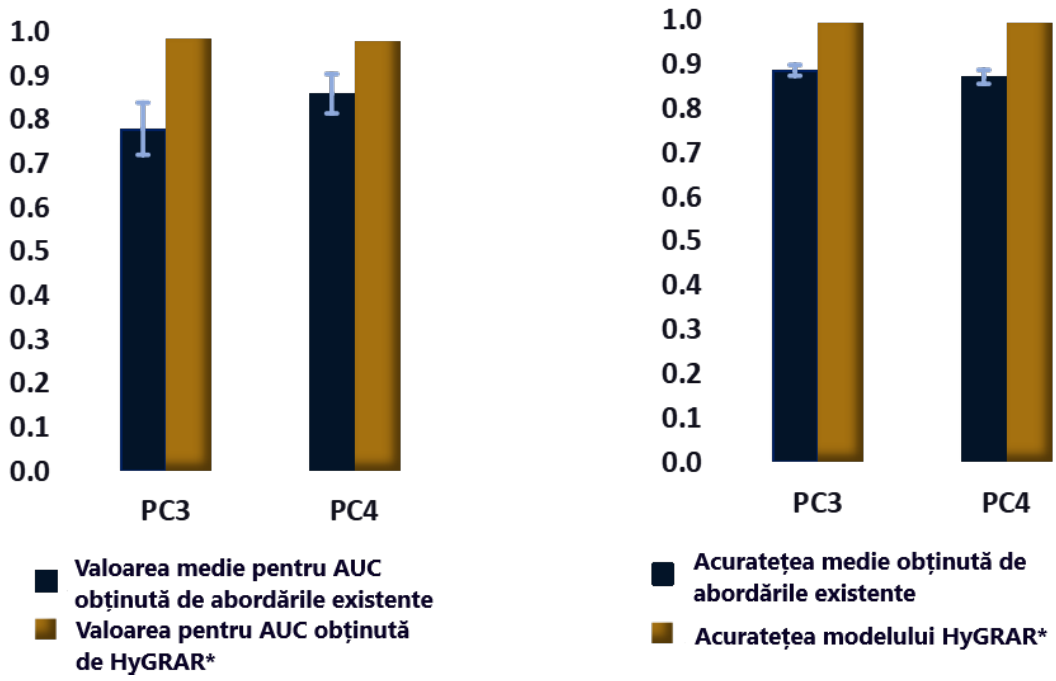


Figure 3.3: Performața de clasificare relativă a modelului HyGRAR*

3.3 Concluzii și direcții viitoare de cercetare

În acest capitol s-a prezentat succint propunerea unui model hibrid de învățare supervizată, numit HyGRAR, ca o abordare a problemei PDS. HyGRAR combină extragerea de GRAR cu RNA. Modelul propus este inovator atât din perspectiva Inginerie Software (IS) cât și a ÎA. Rezultatele experimentale au indicat că HyGRAR surclasează, în general, alte metode existente.

Am propus, de asemenea, o versiune îmbunătățită a modelului HyGRAR, numită HyGRAR*, aceasta înlocuind euristica folosită pentru clasificarea efectivă în versiunea inițială cu o metodologie de clasificare adaptivă, bazată pe MLP, sporind, astfel, performanța.

Performanța excelentă a modelelor HyGRAR și HyGRAR* se datorează, cel mai probabil, complexității și adaptabilității lor. Pe de altă parte, complexitatea afectează interpretabilitatea regulilor de clasificare, însă doar câteva modele simple, cu performanță de clasificare inferioară, o păstrează. O altă limitare indirectă poate fi relevanța insuficientă a metricilor software folosite pentru predicție. Prin urmare, o direcție de cercetare ulterior prin care să fie contracarată această limitare ar fi extragerea automată de caracteristici software relevante. Ca o viitoare direcție adițională, plănuim să continuăm optimizarea clasificatorului HyGRAR*, de pildă prin ponderarea predicțiilor efectuate de MLP cu performanța lor evaluată pe un set de date de validare.

Capitolul 4

Noi abordări pentru Estimarea Cuplării Software

Capitolul curent prezintă contribuțiile originale, bazate pe Învățare Automată (ÎA), la Estimarea Cuplării Software (ECS).

Acestea constau în introducerea unei noi măsuri de cuplare conceptuală [70] și includerea ei într-o nouă măsură de cuplare agregată [32].

Secțiunea 4.1 prezintă abordarea originală constând în definirea unei noi măsuri de cuplare conceptuală, numită ConC, calculată pe baza unor reprezentări învățate nesupervizat ale codului sursă [70]. Măsura ConC este comparată experimental cu măsuri de cuplare existente, inclusiv din perspectiva restructurării software la nivel de pachete.

Secțiunea 4.2 introducere propunerea noastră pentru o măsură de cuplare agregată, numită ACE, care combină cuplarea conceptuală cu cea structurală [32]. Această secțiune detaliază evaluarea experimentală comparativă a măsurii ACE, analiză având ca obiective demonstrarea faptului că ACE diferă de măsurile de cuplare existente și este mai eficientă decât acestea în contextul analizei impactului modificărilor aduse unui proiect software. Secțiunea 4.3 sumarizează capitolul și indică posibile direcții de cercetare viitoare.

Abordările prezentate în acest capitol au fost publicate în [70] și [32].

4.1 ConC: O măsură de Cuplare Conceptuală bazată pe Rețele Neuronale Artificiale

În această secțiune prezentăm propunerea originală constând în definirea cuplării conceptuale dintre două componente software pe baza unor reprezentări învățate nesupervizat ale codului sursă [70]. Aceste reprezentări vectoriale numerice sunt învățate folosind Doc2Vec, modelul bazat pe Rețea Neuronală Artificială (RNA), succint descris în Secțiunea 1.1.2.4.

Scopul evaluării experimentale prezentate în această secțiune constă în a arăta că noua măsură de cuplare conceptuală propusă exprimă alte aspecte ale cuplării software decât cele acoperite de măsurile de cuplare structurale, precum și că acesta este mai eficientă pentru restructurarea proiectelor software la nivel de pachete.

4.1.1 Motivație

Motivația aferentă problemei ECS este prezentată în Secțiunea 1.2.2. În cele cu urmează, motivăm abordarea noastră specifică.

Cuplarea conceptuală [91] măsoară similaritatea conceptuală (sau semantică) dintre două componente software și, conform [9], această este cea mai apropiată de percepția dezvoltatorului

asupra ideii de cuplare.

Deoarece elementele constitutive ale codului sursa, precum identificatorii și comentariile, conțin, de regulă, informații semantice [91], am propus o nouă măsură de cuplare conceptuală, învățată pornind de la codul sursă.

4.1.2 Măsura de cuplare conceptuală propusă (ConC)

Pentru a extrage semantica codului sursă corespunzător unei anumite entități software e, se efectuează pașii prezentați în continuare.

Mai întâi, codul sursă este convertit într-un corpus text incluzând comentariile, identificatorii, etc. Apoi, acest corpus este transformat într-un vector de atribute numerice $ConV(e) = (e^1, e^2, \ldots, e^m)$. Vectorii numerice asociați entităților software sunt învățați folosind Doc2Vec.

În final,cuplarea conceptuală dintre două componente software este calculată ca similaritatea dintre vectorii numerici corespunzători lor.

Definition 6. *Vectorul conceptual* asociat unei entități software e este definit ca reprezentarea acesteia în spațiul Doc2Vec, $ConV(e) = (e^1, e^2, \ldots, e^m)$.

Definition 7. *Cuplarea conceptuală* dintre două entități software e_1 și e_2, notată prin $ConcC(e_1, e_2)$, este definită ca *similaritatea* dintre vectorii conceptuali $({e_1}^1, {e_1}^2, \ldots, {e_1}^m)$ și $({e_2}^1, {e_2}^2, \ldots, {e_2}^m)$ asociați lor.

În definirea măsurii ConC, similaritatea dintre vectorii conceptuali poate fi calculată ca similaritatea euclidiană exprimată în Formula (4.1) sau precum în Formula (4.2), ca similaritate cosinus.

$$ConcC(e_1, e_2) = \frac{1}{1 + \sqrt{\sum_{i=1}^{m} (e_1^i - e_2^i)^2}} \tag{4.1}$$

$$CC(e_1, e_2) = \frac{\left| \sum_{i=1}^{m} ({e_1}^i \cdot {e_2}^i) \right|}{\sqrt{\sum_{i=1}^{m} ({e_1}^i \cdot {e_1}^i)} \cdot \sqrt{\sum_{i=1}^{m} ({e_2}^i \cdot {e_2}^i)}} \tag{4.2}$$

În evaluarea experimentală, am optat pentru similaritatea euclidiană.

4.1.3 Experimente și rezultate

Ca studiu de caz experimental, am considerat sistemul Apache Commons DBUtils, versiunea 1.3.

Experimentele au fost orientate în două direcții.

Pe de o parte, am evidențiat, folosind Analiza Componentelor Principale (PCA) [2], faptul că ConC exprimă noi aspecte ale cuplării, nedetectate de măsurile de cuplare structurală (CBO [21], MPC [59], DAC [59], ICH [57]).

Pe de altă parte, am testat comparativ ConC și măsurile de cuplare structurală în termenii relevanței lor pentru restructurarea software la nivel de pachete. Pentru efectuarea comparației, am folosit Rețele cu Auto-Organizare (SOM), modelul învățând mapări care au evidențiat faptul că considerarea vectorilor conceptuali conduce la o mai bună structură decât folosirea cuplării structurale.

4.1.4 Compararea măsurii ConC cu măsuri existente de cuplare conceptuală

Măsura de cuplare ConC definită în această secțiune se diferențiază de măsurile existente de cuplare conceptuală. În loc de a folosi Indexarea Semantică Latentă (LSI) [42] pentru reprezentarea corpusului text, am folosit Doc2Vec. Doc2Vec este un model bazat pe predicție, în timp ce LSI este un model bazat pe numărare. Mia mult, Doc2Vec este considerat în literatură ca fiind mai capabil de surprinderea semanticii textului decât modelele de regăsire a informației bazate pe numărare [58].

4.2 ACE: O măsură de Cuplare Agregată

În secțiunea curentă, introducem o nouă măsură de cuplare agregată, numită ACE.

Scopul major al analizei experimentale prezentate în această secțiune este de a justifica relevanța agregării cuplărilor structurale și conceptuale în ACE, răspunzând următoarelor întrebări de cercetare:

RQ1 Este ACE capabi;[s[exprime noi aspecte ale cuplării,care nu sunt detectate de măsurile de cuplare structurală și conceptuală existente în literatură?

RQ2 În ce măsură este ACE corelată cu măsurile de cuplare structurală și conceptuală existente în literatură?

RQ3 Este ACE eficientă pentru analiza impactului modificărilor software și cum se compară aceasta cu măsurile similare deja propuse în literatură pentru această activitate?

4.2.1 Motivație

Motivația pentru ECS este dată în Secțiunea 1.2.2. Așadar, în acest secțiune motivăm strict combinarea cuplării structurale cu cea conceptuală, într-o nouă măsură de cuplare agregată.

În primul rând, sunt unele activități ale ingineriei software (de pildă, restructurarea software) pentru care metricile de cuplare structurală, de unele singure, pot fi insuficiente.

În al doilea rând, Bavota și co-autorii [9] au concluzionat pe bază experimentală între cuplare a conceptuală și cea structurală există o relație de complementaritate parțială.

În al treilea rând, combinarea diferitelor metrici software este încurajată în literatură [44, 48].

4.2.2 Măsura de cuplare agregată propusă (ACE)

Componenta structurală a măsurii agregate ACE este o adaptarea a unei măsuri de cuplare generice din literatură [101]. Aceasta este calculată pe baza proprietăților relevante pe care componentele software le au în comun.

Definition 8. Cuplarea structurală dintre două entități software e_1 și e_2 din cadrul unui sistem software S, notată cu $SC(e_1, e_2)$, este dată de următoarea formulă:

$$SC(e_1, e_2) = \frac{|rp(e_1) \cap rp(e_2)|}{|rp(e_1) \cup rp(e_2)|} \tag{4.3}$$

unde $rp(e)$ este o mulțime de *proprietăți relevante* ale unei entități software $e \in S$.

- Dacă e este o metodă, atunci $rp(e)$ conține: metoda în sine, clasa care definește metoda, toate atributele accesate de metodă, toate celelalte metode apelate de e și toate celelalte metode care suprascriu e.

- Dacă e este o clasă, atunci $rp(e)$ contains: the class itself, all the attributes and the methods that are defined in e, all the interfaces which are implemented by e and all the application classes which are extended by class e.

În definirea măsurii de cuplare agregată ACE, măsura de cuplare structurală definită anterior este complementată de măsura de cuplare conceptuală ConC introdusă în Secțiunea 4.1.2. Ca și componentă a măsurii ACE, ConC este definită folosind similaritatea cosinus, spre deosebire de definiția considerată în evaluarea experimentală sumarizată în Secțiunea 4.1.3. Din acest motiv, în secțiunea curentă, folosim o notație distinctă, și anume CC, și îi formulăm în cele ce urmează definiția locală.

Definition 9. Cuplarea conceptuală dintre două entități software e_1 și e_2, notată cu $CC(e_1, e_2)$, este definită ca valoarea absolută a cosinusului dintre vectorii $({e_1}^1, {e_1}^2, \ldots, {e_1}^m)$ și $({e_2}^1, {e_2}^2, \ldots, {e_2}^m)$ corespunzând reprezentărilor lor în spațiul Doc2Vec.

$$CC(e_1, e_2) = \frac{\left|\sum_{i=1}^{m} ({e_1}^i \cdot {e_2}^i)\right|}{\sqrt{\sum_{i=1}^{m} ({e_1}^i \cdot {e_1}^i)} \cdot \sqrt{\sum_{i=1}^{m} ({e_2}^i \cdot {e_2}^i)}} \tag{4.4}$$

Considerând măsurile de cuplare structurală și conceptuală introduse în formulele (4.3) și, respectiv, (4.4), definim măsura de cuplare agregată dintre două entități software ca ca o combinație liniară a similarității lor structurale (SC) și conceptuale (CC).

Definition 10. Cuplarea agregată dintre două entități software e_1 și e_2, notată cu $AC(e_1, e_2)$, este definită ca o combinație liniară între cuplarea lor *structurală* $SC(e_1, e_2)$ și cea *conceptuală* $CC(e_1, e_2)$.

$$AC(e_1, e_2) = w \cdot SC(e_1, e_2) + (1 - w) \cdot CC(e_1, e_2) \tag{4.5}$$

unde w este o pondere subunitară care exprimă importanța relațiilor *structurale* în cadrul măsurii agregate.

Definition 11. Cuplarea agregată a unei entități software e în cadrul unui sistem software $\mathcal{S}$, notată cu $ACE(e)$, este definită ca

$$ACE(e) = \frac{\sum_{\substack{e' \in \mathcal{S} \\ e \neq e'}} AC(e, e')}{p} \tag{4.6}$$

unde $p = |\{e' | e' \in \mathcal{S}, e' \neq e\}|$ este numărul de componente software din $\mathcal{S}$ cu excepția componentei e.

4.2.3 Comparație între ACE și alte măsuri de cuplare

Pentru a răspunde întrebării RQ1, am aplicat SOM vizualizând astfel, comparativ, sistemele software reprezentate folosind diferite măsuri de cuplare, iar pentru a răspunde întrebării RQ2 am efectuat o analiză a corelațiilor.

Ca studii de caz experimentale, ne-am folosit de Common DBUtils, de un proiect care facilitează aplicarea Învățării prin Întărire Și sistemul software Apache Collections.

Aplicarea SOM a dus la mapări diferite, în timp ce analiza de corelații a rezultat în corelații Pearson și Spearman reduse, între ACE și alte măsuri de cuplare, toate acestea confirmând că ACE diferă de măsurile de cuplare existente.

4.2.4 Analiza impactului modificărilor software folosind ACE

4.2.4.1 Seturi de date

Cu scopul principal de a răspunde celei de-a treia întrebări de cercetare, RQ3, am renunțat la proiectul legat de învățarea prin întărire, deoarece dispune de o singură versiune, dar am adăugat toate versiunile disponibile în baza de date SVN pentru DButils și Apache Collections.

Tabelul următor sumarizează conținutul aferent celor două sisteme.

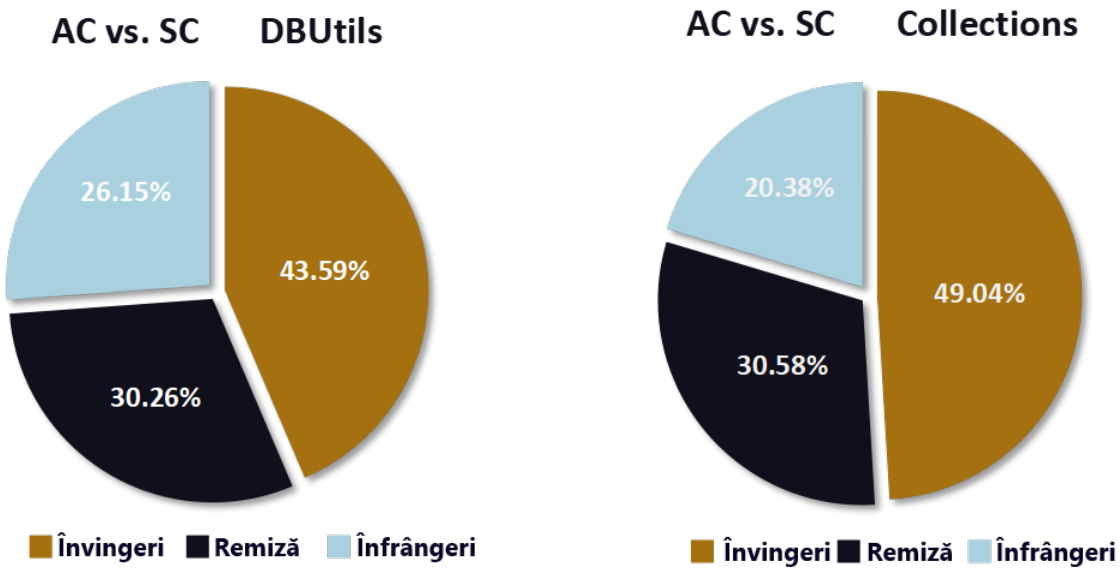
Sistemul software	Numărul versiunilor	Numărul total al commit-urilor	Numărul total al claselor modificate
DbUtils	8	54	366
Apache Collections	10	2953	9026

4.2.4.2 Scopul experimentului

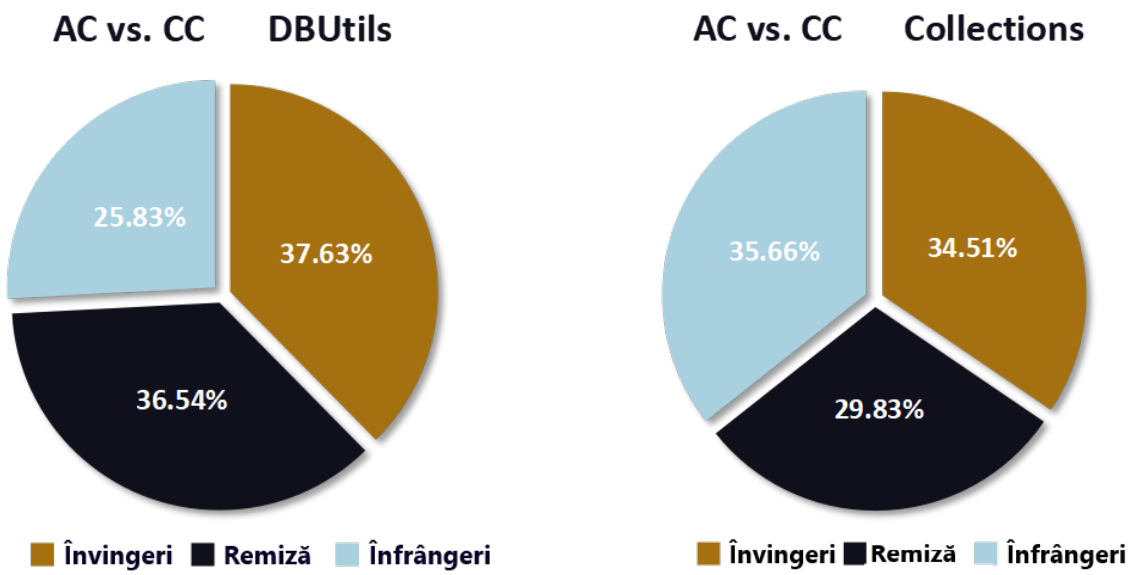
Intuiția practică pe care se fondează experimentul este aceea că analizând un commit, se poate evalua impactul unei modificări. Pe de altă parte, când un dezvoltator software face o modificare, de regulă va căuta componentele afectate pe baza cuplării directe (sau structurale), în toate clasele care definesc sau folosesc metoda modificată. Așadar, scopul se reduce la a arăta că a ghida explorarea în funcție de cuplarea agregată este mai eficient, în sensul că aceasta reduce numărul de clase care trebuie analizate pentru a le identifica pe toate cele afectate.

4.2.5 Rezultate

Per ansamblu, ACE surclasează sau egalează măsurile de cuplare structurală în circa 74% din cazuri, considerând sistemul DBUtils. Pentru Collections, un și mai mare procent în avantajul nostru, unul de aproape 80%.



Am comparat AC și cu măsura de cuplare conceptuală bazată pe LSI [91], în termenii eficacității lor în evaluare impactului modificărilor software. AC a dovedit o performanță superioară în cazul DBUtils, dar a fost ușor depășită de măsura de conceptuală în cazul sistemului Collections.



4.3 Concluzii și direcții viitoare de cercetare

În acest capitol au fost prezentate propunerile noastre pentru noi măsuri de cuplare.

Am abordat ECS introducând o nouă măsura de cuplare conceptuală bazată pe ÎA, numită ConC,și o nouă măsură de cuplare agregată, numită ACE, care combină cuplarea structurală cu cea conceptuală.

Rezultatele experimentale obținute relevă, pe de o parte, că ConC exprimă alte dimensiuni ale cuplării software decât cele surprinse de măsurile de cuplare structurală, fiind, totodată, mai eficientă pentru restructurarea la nivel de pachete iar, pe de alta parte, că ACE surclasează, în general, măsurile clasice care indică dependențe directe, precum și o măsură de cuplare conceptuală înrudită, în a indica entitățile software afectate de o anumită modificare adusă proiectului.

ConC și ACE pot fi folosite pentru proiecte software dezvoltate atât în paradigma programării procedurale, cât și a celei orientate-obiect. Cu toate acestea, experimentele au fost efectuate considerând sisteme orientate-obiect. Prin urmare, o primă direcție de investigare ulterioară ar fi evaluarea experimentală pe alte studii de caz, corespunzând unor paradigme de programare diferite.

Ne propunem, în plus, să investigăm utilizarea unei granularități diferite, la nivel de metodă, în locul celei la nivel de clasă pe care o considerăm în propunerea curentă, precum și noi modalități de agregare a diferitelor măsuri de cuplare.

Capitolul 5

Cuplarea Software ca suport pentru Predicția Defectelor Software

Acest capitol sumarizează contribuțiile originale orientate în direcția utilizării cuplării ca fundament pentru identificarea componentelor software defecte. Aceste contribuții constau în propunerea unui nou model hibrid de predicție a defectelor software, pe baza unor atribute conceptuale învățate automat din codul sursă [73] și introducerea unei noi suite de metrici derivate din cuplarea conceptuală, ca suport ca suport pentru Predicția Defectelor Software (PDS) [77]. Așadar, acest ultim capitol al tezei pune în legătură cele două probleme din domeniul Inginerie Software (IS) abordate, PDS și Estimarea Cuplării Software (ECS).

Secțiunea 5.1 introduce o abordare originală a problemei PDS în care o nouă versiune a modelului de clasificare hibrid care combină extragerea de Reguli de Asociere Relaționale Graduale (GRAR) cu Rețele Neuronale Artificiale (RNA), numită DePSem [73], detectează entitățile software defecte pe baza unor atribute conceptuale învățate din codul sursă. Secțiunea include și o evaluare a modelului DePSem pe 7 seturi de date pentru PDS care corespund unor proiecte software disponibile public.

Noua suită de metrici bazate pe cuplarea conceptuală propusă ca suport pentru PDS și numită COMET [77] este prezentată în Secțiunea 5.2. Suita de metrici este comparată experimental cu metricile software convenționale, în termenii relevanței pentru identificarea componentelor software defecte.

Secțiunea 5.3, ca secțiune finală a capitolului, enunță concluziile și oferă posibile direcții de cercetare viitoare.

Abordările prezentate în acest articol au fost publicate în [73] și [77].

5.1 DePSem: Un model de predicție al defectelor software hibrid, bazat pe atribute conceptuale învățate din codul sursă

Această secțiune deschide investigarea relevanței pe care o are cuplarea conceptuală în evaluarea predispoziției componentelor software față de defecte. Aceasta prezintă un model de clasificare hibrid care combină extragerea de GRAR cu RNA pe care l-am propus pentru detectarea entităților software defecte pe baza unor atribute conceptuale extrase automat din codul sursă [73].

5.1.1 Motivație

Cuplarea software se află în strânsă relație cu calitatea arhitecturii software, iar defectele software sunt efecte ale unei calități slabe, inclusiv a unei arhitecturi precare. Așadar, defectele de proiectare fac un proiect mai susceptibil să fie dobândească defecte [33]. Pe de altă parte, majoritatea defectelor sunt cauzate de neglijarea sau neconștientizarea unor dependențe, cuplarea software indicându-le pe cele tehnice. Prin urmare, este probabilă o relație între cuplare și defectele software. Chiar și așa, există doar foarte puține abordări în literatură care investighează tipuri de cuplare diferite de cea structurală în legătură cu defectele software [18, 52].

În aceste condiții, ne-am propus analizarea interacțiunii dintre cuplarea conceptuală și defectele software. Prin aceasta, și cea de-a doua direcție de cercetare întâlnită în literatura aferentă problemei PDS, și anume proiectarea unor atribute software relevante [108] pentru discriminarea componentelor software defecte, pe lângă propunerea unor algoritmi de clasificare performanți (HyGRAR [75] și HyGRAR* [69]), dar care fundamentează clasificarea pe metrici software clasice, proiectate manual.

5.1.2 Metodologie

DePSem implică doi pași principali: un pas de **colectare a datelor** nesupervizat și un pas de **clasificare** supervizat.

Colectarea datelor presupune transformarea codului sursă aferent entităților software în vectori numerici de atribute conceptual extrase folosind Doc2Vec [56] și Indexarea Semantică Latentă (LSI) [34].

Pasul de **clasificare** se divide în trei faze succesive:

1) **Extragerea** mulțimilor **de GRAR** interesante care discriminează între entitățile software defecte și cele neafectate.

2) **Antrenarea** Perceptroni Multistrat pentru a detecta entiățile defecte pe baza GRAR extrase la pasul 1),

3) **Testarea** unei noi instanțe în sensul de a fi repartizată fie în clasa instanțelor defecte, fie a celor lipsite de defecte, folosind modelul de clasificare rezultat după pasul 2).

Comparând DePSem cu HyGRAR și HyGRAR*, DePSem utilizează atribute semantice extrase automat din codul sursă, în timp ce ceilalți doi clasificatori se bazează pe metrici software convenționale, folosite la scară largă. Pe de altă parte, RNA nu sunt folosite aici pentru a învăța relații graduale - s-a optat pentru predefinirea lor - dar sunt folosite pentru a învăța regula de clasificare, precum în cazul versiunii HyGRAR*.

5.1.3 Experimente și rezultate

5.1.3.1 Seturi de date

Pentru evaluarea DePSem am folosit 7 seturi de date (JEdit 3.2, JEdit 4.0, JEdit 4.1, JEdit 4.2, JEdit 4.3, Ant 1.7 și Tomcat 6.0) corespunzând unor sisteme software orientate-obiect.

Pentru a pregăti seturile de date pentru experimentele noastre, am transformat numărul de defecte într-o etichetă binară indicând, pentru o componentă software, dacă aceasta este defectă sau nu și am înlocuit metricile software cu atributele semantice învățare parcursul pasului inițial de colectare a datelor, din codul sursă pre-procesat minimal.

5.1.3.2 Evaluare

Pentru testarea performanței de clasificare a modelului DePSem, s-a folosit aceeași metodologie de evaluare precum în cazul celorlalte modele hibride. Pe lângă aceasta, am recurs la vizualizarea datelor, reprezentate ca vectori conceptuali, folosindu-ne de tehnica t-SNE.

5.1.3.3 Rezultate

Figurile 5.1, 5.2, 5.3 și 5.4 corespund vizualizărilor t-SNE [106] ale datelor corespunzând sistemelor Ant și Tomcat și versiunilor 3.2 și 4.3 ale sistemului JEdit, reprezentate cu ajutorul modelului Doc2Vec.

Este interesant de observat faptul că multe clase defecte, colorate cu galben, sunt poziționate în aceleași regiuni mai rarefiate, în timp ce în porțiunile foarte aglomerate nu există sau există foarte puține clase defecte. Aceasta ar putea indica faptul că entitățile software care sunt puternic culate cu alte entități de asemenea puternic cuplate între ele sunt cu o mai mică probabilitate defecte. Am identificat suport pentru această observație și în literatură [18], iar intuiția este că pe măsură ce componente software inter-cuplate sunt dezvoltate, dezvoltatorii devin conștienți de ele, astfel încât știu cum se propagă modificările și unde să intervină astfel încât să reducă probabilitatea de a introduce defecte.

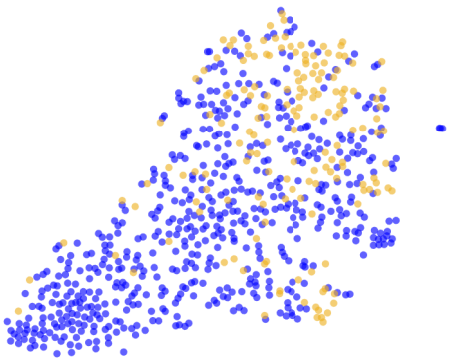


Figure 5.1: Reprezentarea t-SNE a sistemului Ant 1.7 folosind vectorii conceptuali învățați de Doc2Vec

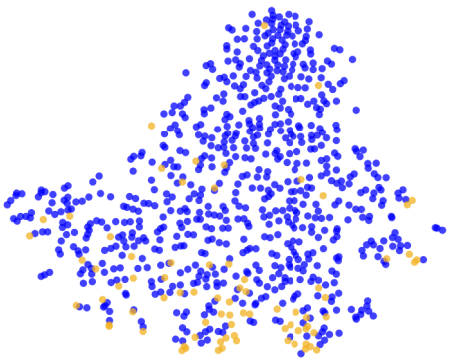


Figure 5.2: Reprezentarea t-SNE a sistemului Tomcat 6.0 folosind vectorii conceptuali învățați de Doc2Vec

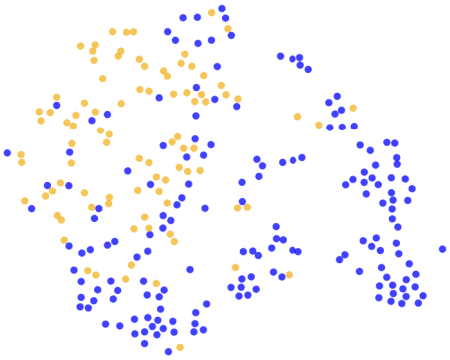


Figure 5.3: Reprezentarea t-SNE a sistemului JEdit 3.2 folosind vectorii conceptuali învățați de Doc2Vec

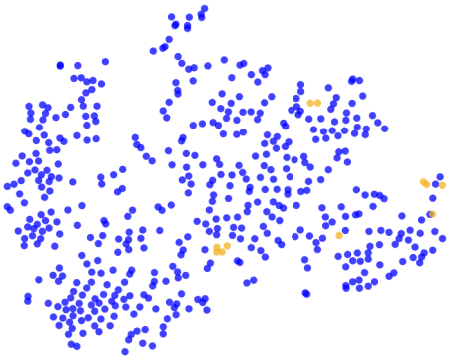


Figure 5.4: Reprezentarea t-SNE a sistemului JEdit 4.3 folosind vectorii conceptuali învățați de Doc2Vec

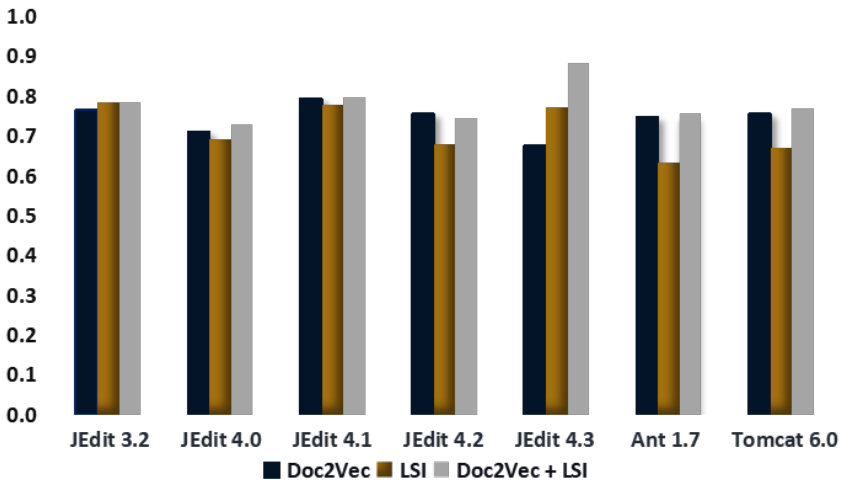


Figure 5.5: Valori AUC obținute folosind Doc2Vec, LSI sau ambele, combinate

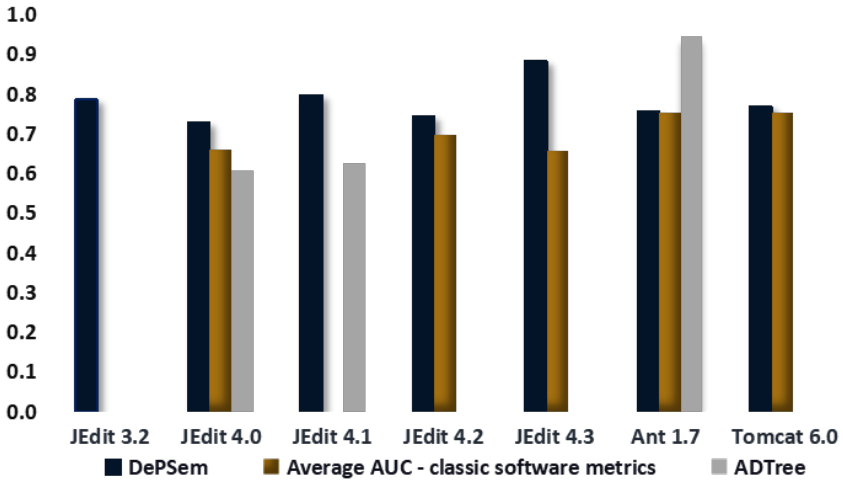


Figure 5.6: Comparația performanței modelului DePSem cu cea a abordărilor înrudite

Comparând folosirea doar a modelului Doc2Vec cu a modelului LSI, exclusiv, sau cu a ambelor modele, pentru extragerea atributelor conceptuale, după cum se poate observa în Figura 5.5, reprezentarea combinată a condus la cele mai mari valori pentru Aria de sub curba ROC (AUC), pentru majoritatea sistemelor software considerate.

5.1.4 Comparație cu abordări înrudite

Literatura de specialitate conține numeroase abordări pentru detectarea defectelor software, însă majoritatea acestora se bazează pe folosirea metricilor convenționale, inclusiv a unora care exprimă relațiile structurale dintre componentele software. Există doar câteva abordări care implică alte tipuri de cuplare [18, 52, 51]. Acestea sunt înrudite cu DePSem doar în sensul în care valorifică dependențe diferite de cele surprinse de măsurile de cuplare structurale, folosite la scară largă. Principala diferență constă în faptul că, în cazul DePSem, atributele semantice sunt extrase automat din codul sursă, pe când măsurile de cuplare logică, de pildă, sunt proiectate manual, pe baza datelor istorice.

După cum este reprezentat în Figura 5.6, comparat cu predictorii de defecte bazați pe metri-

cile convenționale [75], DePSem surclasează valoarea medie pentru AUC obținută de aceștia, în cazul tuturor sistemelor. DePSem surclasează, de asemenea, în cazul sistemelor JEdit 4.0 și JEdit 4.1, abordarea bazată pe caracteristici semantice extrase din Arborii Sintactici Abstracți (AST) ai programelor, introdusă în [108]. Menționăm că, spre deosebire de această abordare [108], noi nu am redus dificultatea seturilor de date prin a elimina instanțe cu etichete posibil incorecte, iar barele absente din Figura 5.6 se datorează faptului că nu au fost raportate rezultate în cazul respectivelor seturi de date.

5.2 COMET: O suită de metrici derivate din cuplare pentru predicția defectelor software

Această secțiune prezintă succint noua suită de metrici derivată din cuplarea conceptuală pe care am propus-o ca suport pentru PDS.

5.2.1 Motivație

Alegerea metricilor software care să fie folosite ca atribute pentru PDS s-a dovedit a avea un impact mai puternic asupra performanței de clasificare decât selectarea tehnicii de clasificare, cel puțin în cazul modelelor de clasificare simple, convenționale [92]. Totuși, majoritatea modelelor de PDS se bazează pe metricile software [96], acestea fiind folosite la scară largă, dar, de asemenea, criticate [41]. Având în vedere impactul avut de selectarea caracteristicilor software pe baza cărora să se efectueze predicția, precum și rezultatele experimentale încurajatoare prezentate în secțiunea anterioară, propunem noi metrici software, bazate pe cuplarea conceptuală, pentru a facilita PDS.

5.2.2 Suita de metrici COMET

Pentru definirea suitei de metrici COMET, se efectuează următorii pași:

1. Mai întâi, folosind Doc2Vec [56] și LSI [20] se extrag automat, nesupervizat, din codul sursă, atribute software conceptuale.

2. Cel de-al doilea pas constă în calculul cuplării conceptuale pentru fiecare pereche de entități software, ca similaritate între vectorii conceptuali corespunzători lor. Pentru aceasta, se folosește atât similaritate *cosinus*, cât și cea *euclidiană*.

3. Pentru fiecare entitate software, se construiește câte o secvență numerică considerând cuplarea conceptuală a entității în raport cu: (1) toate celelalte entități din sistemul software; (2) toate celelalte entități *defecte* și (3) toate celelalte entități *lipsite de defecte*.

4. Pasul final constă în definirea metricilor software compunând suita COMET considerând *media*, *valoarea maximă* și *deviația standard*, ca măsuri ale statisticii descriptive, aplicându-le secvențelor construite la pasul 3 pentru fiecare entitate software în parte.

Urmând metodologia descrisă anterior (prin pașii 1-4), se obțin 36 ($2 \cdot 2 \cdot 3 \cdot 3$) metrici software care compun suita COMET.

5.2.3 Experimente și rezultate

Experimentele își propun compararea metricilor COMET cu metricile software Promise [96].

5.2.3.1 Seturi de date

Experimentele au fost efectuate pe aceleași 7 sisteme software care au fost folosite pentru evaluarea modelului DePSem, însă for pentru fiecare dintre seturile de date Promise, am construit câte un set de date adițional în care fiecare entitate software din setul de date original este reprezentată ca un vector numeric de lungime 36, ale cărui componente sunt metricile COMET.

5.2.3.2 Analiza bazată pe corelații

Am început evaluarea relevanței metricilor COMET pentru PDS, în comparație cu cea a metricilor Promise, printr-o analiză a corelațiilor dintre metricile COMET versus metricile Promise și prezența defectelor software.

Analiza a evidențiat faptul că, în cazul seturilor de date considerate pentru experimentare, metricile COMET sunt mai corelate cu predispoziția la defecte decât metricile clasice Promise. Corelația Pearson dintre metricile COMET și defectuozitatea entităților software este cu 19.67% mai mare decât cea dintre metricile Promise și prezența defectelor (0.238 versus 0.199). În cazul corelației Spearman, acest procent relativ este de 7.34% (0.237 versus 0.221).

5.2.3.3 Analiza bazată de învățare nesupervizată

Mai apoi, am comparat suita COMET cu setul de metrici Promise, în termenii relevanței pentru PDS, folosindu-ne de învățarea nesupervizată. Pentru aceasta, am utilizat tehnica t-SNE [106]. Nu ne-am limitat la o comparație strict vizuală, ci am efectuat și o comparație numerică bazată pe o măsură de dificultate a seturilor de date generate de reprezentarea în spațiul t-SNE.

Dificultatea generală de clasificare [115] este redusă în 6 dintre cele 7 studii de caz prin înlocuirea metricilor Promise cu metricile COMET, în timp ce dificultatea distingerii componentelor defecte de celelalte este redusa semnificativ pentru toate studiile de caz. Aceste rezultate numerice confirmă că, din analiza nesupervizată folosind t-SNE, reiese că suita de metrici COMET este mai relevantă decât setul de metrici Promise pentru deosebirea între componentele software defecte și celelalte.

5.2.3.4 Analiza bazată pe învățare supervizată

În final, am completat comparația dintre suita COMET și metricile Promise printr-un studiu experimental efectuat dintr-o perspectivă supervizată. Pentru aceasta, am selectat 3 modele de Învățare Automată (ÎA) pentru clasificare supervizată: Regresia Logistică, Cei Mai Apropiați k Vecini (kNN) și o Rețea Neuronală Artificială (RNA). Ca metodologie de evaluare, am aplicat LOO.

Global, folosind cele 36 de metrici COMET în locul celor 20 metrici Promise, ca atribute pentru PDS, valoarea obținută pentru AUC a crescut cu mai mult de 11%.

Așadar, comparația bazată pe învățare supervizată a consolidat concluzia că metricile COMET sunt relevante pentru PDS, surclasând, pentru cele 7 studii de caz considerate, metricile Promise folosite la scară largă.

5.2.4 Comparație cu abordări înrudite

Figura 5.7 compară performanța clasificatorilor folosind metricile COMET cu cea obținută în literatură folosind metricile convenționale și cu cea a modelului original DePSem.

Comparația reliefează faptul că performanța obținută folosind metricile COMET depășește atât performanța modelului DePSem, cât și performanța medie a abordărilor care folosesc metricile Promise.

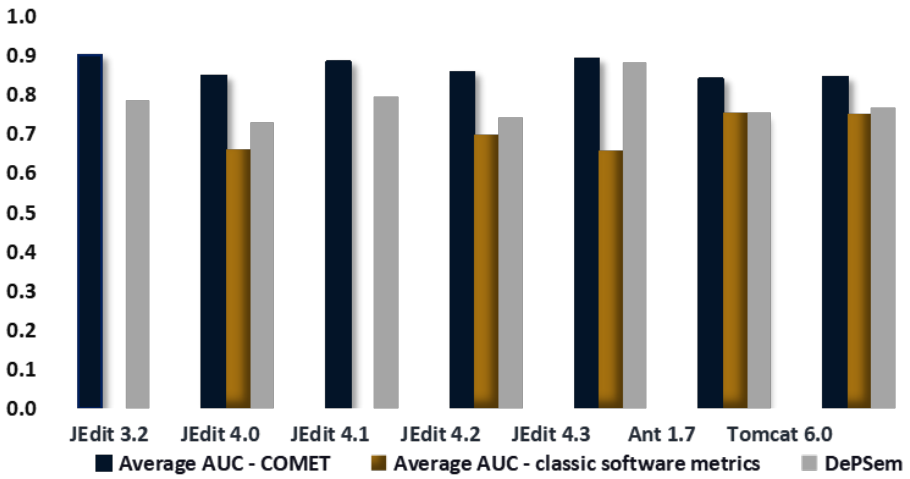


Figure 5.7: Comparație cu abordări înrudite

5.3 Concluzii și direcții viitoare de cercetare

Acest capitol a prezentat succint abordările noastre pentru PDS bazate pe caracteristici conceptuale extrase automat din codul sursă și o suită de metrici derivate din cuplarea conceptuală, de asemenea pentru PDS. Ambele contribuții au avut ca scop confirmarea relevanței pe care cuplarea conceptuală o are pentru predicția componentelor software defecte.

Experimentele efectuate au evidențiat influența pe care cuplarea conceptuală o are asupra predispoziției componentelor software față de defecte.

Metricile derivate din cuplarea conceptuală au fost comparate cu metrici software arhicunoscute pentru PDS, din trei perspective diferite: a unei analize de corelații, a învățării nesupervizate și a învățării supervizate. Toate aceste comparații au semnalat faptul că metricile derivate din cuplarea conceptuală sun superioare, din punct de vedere al performanței de predicție, metricilor software standard, folosite la scară largă în literatura specializată pe PDS.

Ca o primă direcție de investigare viitoare, intenționăm să extindem evaluare experimentală considerând granularități diferite (de pildă, predicția defectelor la nivel de metodă) și limbaje de programare adiționale. Ca o a doua direcție, ne propunem extinderea suitei COMET astfel încât să includă și cuplarea structurală. Motivația acestei direcții secundare de cercetare este dată de complementaritatea cuplărilor structurală și conceptuală [18], în timp ce cuplarea agregată [32] s-a dovedit a fi eficientă pentru exprimarea dependențelor existente între componentele software.

Concluzii

În această teză au fost prezentate contribuțiile noastre conceptuale aduse domeniului Învățare Automată (ÎA) împreună cu cele aduse cercetării aplicate în Inginerie Software (IS).

Am abordat două probleme din domeniul IS, și anume Predicția Defectelor Software (PDS) și Estimarea Cuplării Software (ECS), ambele fiind esențiale pentru asigurarea calității software. Cuplarea software, de pildă, este relevantă pentru restructurarea software și analiza impactului modificărilor software, facilitând, astfel, o bună structurare și înțelegere a sistemului. În acest sens, am introdus noi măsuri de cuplare conceptuală și agregată, ambele bazate pe ÎA. Mai mult, cuplarea poate afecta predispoziția la defecte a componentelor software, ceea ce noi am confirmat prin a propune cu succes noi atribute bazate pe cuplare care să fie folosite ca suport pentru PDS. Prin aceasta, am urmat una dintre cele două direcții principale din domeniul de cercetare foarte activ aferent problemei PDS, aceasta fiind propunerea unor noi atribute software relevante. Cea de-a doua direcție constă în dezvoltarea unor modele de clasificare noi și îmbunătățite, noi urmând-o și pe aceasta, prin propunerea unor noi abordări hibride care combină extragerea Reguli de Asociere Relaționale Graduale (GRAR) cu Rețele Neuronale Artificiale (RNA).

Abordările noastre hibride sunt, în același timp, contribuții originale fundamentale aduse domeniului ÎA, precum sunt și introducerea tehnicii de extragere a GRAR, împreună cu versiunile adaptive și dinamice ale algoritmului de extragere. Extragerea GRAR generalizează extragere Reguli de Asociere Relaționale (RAR) îmbogățind-o cu mai o expresivitate superioare și cu stabilitate mărită la erori în date, în timp de versiunile adaptive și dinamice ale algoritmului de extragere a GRAR fac procesul de descoperire a GRAR semnificativ mai eficient în contextul analizării seturilor de date dinamice. Procesul de extragere al GRAR este integrat în modele hibride originale de clasificare care implică, pe lângă acesta, și RNA pentru a învăța relații graduale (HyGRAR și HyGRAR*) și / sau să clasifice o instanță pe baza GRAR interesante care caracterizează clasele (HyGRAR* și DePSem).

Precum am menționat deja, am valorificat contribuțiile fundamentale originale astfel încât să aducem contribuții și cercetării aplicative în domeniul IS. Astfel, mai întâi am abordat problema PDS aplicând modelele hibride HyGRAR și HyGRAR* pentru predicția defectelor software, atât în cadrul aceluiași proiect, cât și într-un cadru generalizat. Acestea au dovedit o performanță de clasificare excelentă atât în cazul sistemelor dezvoltate în paradigma orientată-obiect, cât și al celor dezvoltate în paradigma procedurală. Apoi, am abordat ECS prin a defini noi măsuri de cuplare conceptuală și agregată, ambele bazate pe ÎA, validate după criteriul utilității lor pentru restructurarea software și analiza modificărilor software. În cele din urmă, am pus în legătură PDS și ECS prin utilizarea cu succes a caracteristicilor software bazate pe cuplarea conceptuală ca fundamente pentru predicția componentelor software având defecte, implicând în acesta, deopotrivă, un model hibrid original care folosește procesul de extragerea la GRAR interesante în conjuncție cu RNA (DePSem).

În concluzie, rezultatele cercetării prezentate în această teză au adus contribuții atât domeniului ÎA, cât și domeniului IS. Aplicabilitatea contribuțiilor noastre conceptuale nu este limitată la aria aferentă IS, însă, deocamdată ea a fost dovedită, empiric, în acest cadru. Dintr-o perspectivă diferită, am arătat faptul că domeniul IS poate beneficia de contribuțiile noastre fundamentale, în particular de avantajele generate de reunirea tehnicii de extragere a GRAR cu RNA sub forma

unor modele hibride de ÎA.

Problemele din domeniul IS pe care le-am abordat, în principal PDS și ECS, dar și restructurarea software și analiza impactului modificărilor software, în raport cu care am investigat aplicabilitatea măsurilor de cuplare nou propuse, pot fi încadrate într-o problemă mai generală, și anume evaluarea calității software, aceasta fiind esențială pentru a asigura produse software de calitate superioară. Centrul contribuțiilor noastre în domeniul IS rămâne, totuși, PDS deoarece, în cele din urmă, atât modelele originale hibride de ÎA cât și cuplarea software au fost folosite în predicția defectelor software.

În completarea direcțiilor de investigare viitoare indicate, separat, pentru fiecare capitol în parte, identificăm alte direcții, mai generale, care ar putea influența pozitiv calitatea software, în special prin PDS. Le prezentăm în cele ce urmează.

O primă direcție de investigare ar fi definirea unor noi măsuri de coeziune software, bazate pe tehnici de ÎA și evaluarea relevanței acestora în aprecierea calității software, inclusiv prin PDS. De exemplu, ne propunem să definim măsuri de coeziune conceptuală bazate pe reprezentările furnizate de Doc2Vec pentru codul sursă și să le includem, alături de măsuri de coeziune structurală, într-o măsură de coeziune agregată.

Deoarece cuplarea și coeziunea sunt complementare în evaluarea calității arhitecturii software, o a doua idee de cercetare ulterioară ar fi să le reunim puterea prin dezvoltarea unei noi suite extinse de atribute software relevante pentru PDS, compuse din noi metrici derivate din cuplarea agregată și coeziunea agregată a componentelor software.

Cu privire la modelele de clasificare utilizate pentru predicția defectelor, analize viitoare ar putea explora aplicarea modelului HyGRAR într-un scenariu de detecție a anomaliilor [94], care este potrivit în cazul seturilor de date debalansate. Acesta ar implica învățarea doar a GRAR care caracterizează clasa majoritară (în cazul nostru, a componentelor software lipsite de defecte) și efectuarea clasificării exclusiv pe baza lor.

Mai mult, pentru adaptarea la caracterul dinamic al procesului de dezvoltare software a soluțiilor noastre, ne propunem să incorporăm algoritmul Algoritm Dinamic de extragere al Regulilor de Asociere Relaționale Graduale (DynGRAR) în HyGRAR. Astfel, mulțimea GRAR interesante pe care se bazează clasificarea ar fi adaptată dinamic și eficient pe măsură ce sistemul software evoluează.

Bibliografie

[1] Abaei, G., Rezaei, Z., and Selamat, A. Fault prediction by utilizing self-organizing map and threshold. In *Proceedings of the IEEE International Conference on Control System, Computing and Engineering - ICCSCE* (2013), pp. 465–470.

[2] Abdi, H., and Williams, L. J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics 2*, 4 (2010), 433–459.

[3] AFZAL, W., TORKAR, R., and FELDT, R. Resampling methods in software quality classification. *International Journal of Software Engineering and Knowledge Engineering 22*, 02 (2012), 203–223.

[4] Agarwal, S. *Data mining: Data mining concepts and techniques*. Morgan Kaufmann Publishers Inc., 2014.

[5] Agrawal, R., and Srikant, R. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases* (1994), Morgan Kaufmann Publishers Inc., pp. 487–499.

[6] Ajienka, N., and Capiluppi, A. Understanding the interplay between the logical and structural coupling of software classes. *Journal of Systems and Software 134*, Supplement C (2017), 120–137.

[7] Akbarinasaji, S., Soltanifar, B., Çağlayan, B., Bener, A. B., Miranskyy, A., Filiz, A., Kramer, B. M., and Tosun, A. A metric suite proposal for logical dependency. In *Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics - WETSoM* (2016), ACM, pp. 57–63.

[8] Arisholm, E., Briand, L. C., and Foyen, A. Dynamic coupling measurement for object-oriented software. *IEEE Transactions on Software Engineering 30*, 8 (2004), 491–506.

[9] Bavota, G., Dit, B., Oliveto, R., Di Penta, M., Poshyvanyk, D., and De Lucia, A. An empirical study on the developers perception of software coupling. In *Proceedings of the 2013 International Conference on Software Engineering* (2013), IEEE Press, pp. 692–701.

[10] Bishnu, P., and Bhattacherjee, V. Software fault prediction using quad tree-based k-means clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering 24*, 6 (2012), 1146–1150.

[11] Borzecka, H. Multi-criteria decision making using fuzzy preference relations. *Operations Research and Decisions 3* (2012), 5–21.

[12] Briand, L., Wust, J., and Lounis, H. Using coupling measurement for impact analysis in object-oriented systems. In *Proceedings of IEEE International Conference on Software Maintenance - ICSM. 'Software Maintenance for Business Change'* (1999), IEEE Computer Society, pp. 475–482.

[13] Briand, L. C., and Daly, J. W. A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering 25*, 1 (1999), 91–121.

[14] Broomhead, D. S., and Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex Systems 2* (1988), 321–355.

[15] Campan, A., Serban, G., and Marcus, A. Relational association rules and error detection. *Studia Universitatis Babes-Bolyai Informatica LI*, 1 (2006), 31–36.

[16] Canfora, G., Lucia, A. D., Penta, M. D., Oliveto, R., Panichella, A., and Panichella, S. Multi-objective cross-project defect prediction. In *Proceedings of the 6th International Conference on Software Testing, Verification and Validation* (2013), pp. 252–261.

[17] Catal, C., Sevim, U., and Diri, B. Software fault prediction of unlabeled program modules. In *Proceedings of the World Congress on Engineering* (2009), pp. 1–6.

[18] Cataldo, M., Mockus, A., Roberts, J. A., and Herbsleb, J. D. Software dependencies, work dependencies, and their impact on failures. *IEEE Transactions on Software Engineering 35*, 6 (2009), 864–878.

[19] Chang, H. Y., Lin, J. C., Cheng, M. L., and Huang, S. C. A novel incremental data mining algorithm based on FP-growth for big data. In *Proceedings of the International Conference on Networking and Network Applications - NaNA* (2016), pp. 375–378.

[20] Chen, H., Martin, B., Daimon, C., and Maudsley, S. Effective use of latent semantic indexing and computational linguistics in biological and biomedical applications. *Frontiers in Physiology 4* (2013), 8.

[21] Chidamber, S. R., and Kemerer, C. F. Towards a metrics suite for object oriented design. *Special Interest Group on Programming Languages Notices 26*, 11 (1991), 197–211.

[22] Clark, B., and Zubrow, D. How good is a software: a review on defect prediction techniques. In *Software Engineering Symposium* (Carneige Mellon University, 2001), pp. 1–35.

[23] Cojocar, G. S., and Şerban, G. On some criteria for comparing aspect mining techniques. In *Proceedings of the 3rd Workshop on Linking Aspect Technology and Evolution - LATE* (2007), Association for Computing Machinery, p. 1–5.

[24] Czibula, G., Bocicor, M. I., and Czibula, I. G. Promoter sequences prediction using relational association rule mining. *Evolutionary Bioinformatics*, 8 (2012), 181–196.

[25] Czibula, G., Czibula, I. G., Miholca, D.-L., and Crivei, L. M. A novel concurrent relational association rule mining approach. *Expert Systems with Applications 125* (2019), 142 – 156.

[26] Czibula, G., Czibula, I. G., Sîrbu, A.-M., and Mircea, I.-G. A novel approach to adaptive relational association rule mining. *Applied Soft Computing 36* (2015), 519–533.

[27] Czibula, G., Marian, Z., and Czibula, I. G. Software defect prediction using relational association rule mining. *Information Sciences 264* (2014), 260 – 278.

[28] Czibula, G., Marian, Z., and Czibula, I. G. Detecting software design defects using relational association rule mining. *Knowledge and Information Systems 42*, 3 (2015), 545–577.

[29] Czibula, I., Czibula, G., Miholca, D., and Marian, Z. Identifying hidden dependencies in software systems. *Studia Universitatis Babeș-Bolyai Informatica 62*, 1 (2017), 90–106.

[30] Czibula, I. G., and Czibula, G. Improving systems design using a clustering approach. *IJCSNS International Journal of Computer Science and Network Security 6*, 12 (2006), 40–49.

[31] Czibula, I. G., Czibula, G., and Miholca, D. L. Enhancing relational association rules with gradualness. *International Journal of Innovative Computing, Information and Control 13*, 1 (2017), 289–305.

[32] Czibula, I. G. I., Czibula, G., Miholca, D.-L. D.-L., and Onet-Marian, Z. An aggregated coupling measure for the analysis of object-oriented software systems. *Journal of Systems and Software 148* (2019), 1–20.

[33] D'Ambros, M., Bacchelli, A., and Lanza, M. On the impact of design flaws on software defects. In *Proceedings of the 10th International Conference on Quality Software* (2010), pp. 23–31.

[34] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41* (1990), 391–407.

[35] Dhanabhakyam, M., and M., P. An efficient market basket analysis based on adaptive association rule mining with faster rule generation algorithm. *The SIJ Transactions on Computer Science Engineering and its Applications - CSEA 1*, 3 (2013), 1–6.

[36] e Abreu, F. B. The MOOD metrics set. In *Proceedings of the European Conference on Object-Oriented Programming - ECOOP* (1995), pp. 150–152.

[37] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters 27*, 8 (2006), 861–874.

[38] Francisco, L., Armando, B., Fernando, G., Carlos, C., and Antonio, M. Fuzzy association rules for biological data analysis: A case study on yeast. *BMC Bioinformatics 9*, 1 (2008), 107.

[39] Gagolewski, M., and Lasek, J. The use of fuzzy relations in the assessment of information resources producers' performance. In *Advances in Intelligent Systems and Computing* (2015), vol. 323, pp. 289–300.

[40] Gosain, A., and Bhugra, M. A comprehensive survey of association rules on quantitative data in data mining. In *Proceedings of the 2013 IEEE Conference on Information and Communication Technologies - ICT* (2013), pp. 1003–1008.

[41] Grady, R. B. *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall Press, 1992.

[42] Grossman, D. A., and Frieder, O. *Information Retrieval: Algorithms and Heuristics*, vol. 461. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[43] Hall, T., Beecham, S., Bowes, D., Gray, D., and Counsell, S. A systematic review of fault prediction performance in software engineering. *IEEE Transactions on Software Engineering 38*, 6 (2011), 1276–1304.

[44] Harman, M., McMinn, P., de Souza, J. T., and Yoo, S. Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical Software Engineering and Verification: International Summer Schools, LASER 2008-2010, Revised Tutorial Lectures* (2012), Springer, pp. 1–59.

[45] Hryszko, J., and Madeyski, L. Cost effectiveness of software defect prediction in an industrial project. *Foundations of Computing and Decision Sciences 43*, 1 (2018), 7 – 35.

[46] hua Chang, R., Mu, X., and Zhang, L. Software defect prediction using non-negative matrix factorization. *Journal of Software - JSW 6*, 11 (2011), 2114–2120.

[47] Hüllermeier, E. Association rules for expressing gradual dependencies. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery* (2002), pp. 200–211.

[48] Kagdi, H., Gethers, M., Poshyvanyk, D., and Collard, M. L. Blending conceptual and evolutionary couplings to support change impact analysis in source code. In *2010 17th Working Conference on Reverse Engineering* (2010), pp. 119–128.

[49] Kaski, S., and Kohonen, T. Structures of welfare and poverty in the world discovered by self-organizing maps. In *Neural Networks in Financial Engineering. Proceedings of the Third International Conference on Neural Networks in the Capital Markets* (1996), World Scientific, pp. 498–507.

[50] Khanesar, M. A., and Teshnehlab, M. Direct fuzzy model reference controller for siso nonlinear plants using observer. *International Journal of Innovative Computing, Information and Control 6*, 1 (2010), 297–306.

[51] Kirbas, S., Caglayan, B., Hall, T., Counsell, S., Bowes, D., Sen, A., and Bener, A. The relationship between evolutionary coupling and defects in large industrial software. *Software: Evolution and Process 29*, 4 (2017), 1–19.

[52] Kirbas, S., Sen, A., Caglayan, B., Bener, A., and Mahmutogullari, R. The effect of evolutionary coupling on software defects: An industrial case study on a legacy system. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (2014), ACM, pp. 1–7.

[53] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics 43*, 1 (1982), 59–69.

[54] Ksantini, M., Hammami, M. A., and Delmotte, F. On the global exponential stabilization of Takagi-Sugeno fuzzy uncertain systems. *International Journal of Innovative Computing, Information and Control 11*, 1 (2015), 281–284.

[55] Lampinen, J., and Oja, E. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision 2*, 2-3 (1992), 261–272.

[56] Le, Q. V., and Mikolov, T. Distributed representations of sentences and documents. *Computing Research Repository (CoRR) abs/1405.4* (2014), 1–9.

[57] Lee, Y. S., Liang, B. S., Wu, S. F., and Wang, F. J. Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow. In *Proceedings of the International Conference of Software Quality - ICSQ* (1995), pp. 81–90.

[58] Levy, O., Goldberg, Y., and Dagan, I. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics - TACL 3* (2015), 211–225.

[59] Li, W., and Henry, S. OO metrics which predict maintainability. *Journal of Systems and Software 23*, 2 (1993), 111–122.

[60] Li, Y., Zhang, Z.-H., Chen, W.-B., and Min, F. TDUP: an approach to incremental mining of frequent itemsets with three-way-decision pattern updating. *International Journal of Machine Learning and Cybernetics 8*, 2 (2017), 441–453.

[61] Liu, Y., and Milanova, A. Static analysis for dynamic coupling measures. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research - CASCON* (2006), pp. 10–21.

[62] L.P.F.a, G., de Carvalho A.C.P.L.F.a, and A.C.b, L. Effect of label noise in the complexity of classification problems. *Neurocomputing 160* (2015), 108–119.

[63] Ma, B., Dejaeger, K., Vanthienen, J., and Baesens, B. Software defect prediction based on association rule classification. In *Proceedings of the 2010 International Conference on E-Business Intelligence* (2010), Atlantis Press, pp. 396–402.

[64] Malhotra, R. A defect prediction model for open source software. In *Proceedings of the World Congress on Engineering* (2012), vol. II, pp. 1–5.

[65] Malhotra, R. Comparative analysis of statistical and machine learning methods for predicting faulty modules. *Applied Soft Computing 21* (2014), 286–297.

[66] Marcus, A., Maletic, J. I., and Lin, K.-I. Ordinal association rules for error identification in data sets. In *Proceedings of the tenth International Conference on Information and Knowledge Management - CIKM* (2001), ACM, pp. 589–613.

[67] Marian, Z., Mircea, I., Czibula, I., and Czibula, G. A novel approach for software defect prediction using fuzzy decision trees. In *Proceedings of the 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing - SYNASC* (2016), pp. 240–247.

[68] Marin, M., Deursen, A. V. A. N., and Moonen, L. Identifying crosscutting concerns using fan-in analysis. *ACM Transactions on Software Engineering and Methodology 17*, 1 (2007), 1–37.

[69] Miholca, D. An improved approach to software defect prediction using a hybrid machine learning model. In *Proceedings of the International Symposium on Symbolic and Numeric Algorithms for Scientific Computing - SYNASC* (2018), pp. 443–448.

[70] Miholca, D., Czibula, G., Zsuzsanna, M., and Czibula, I. G. An unsupervised learning based conceptual coupling measure. In *Proceedings of the 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing - SYNASC* (2017), pp. 247–254.

[71] Miholca, D. L. An adaptive gradual relational association rules mining approach. *Studia Universitatis Babes-Bolyai Series Informatica 63*, 1 (2018), 94–110.

[72] Miholca, D.-L., and Czibula, G. DynGRAR: A dynamic approach to mining gradual relational association rules. In *Proceedings of the 23th Knowledge-Based and Intelligent Information and Engineering Systems - KES* (2019), pp. 10 – 19.

[73] Miholca, D.-L., and Czibula, G. Software defect prediction using a hybrid model based on semantic features learned from the source code. In *Proceedings of the International Conference on Knowledge Science, Engineering and Management - KSEM* (2019), Springer International Publishing, pp. 262–274.

[74] Miholca, D.-L., Czibula, G., and Crivei, L. M. A new incremental relational association rules mining approach. In *Proceedings of the 22nd Knowledge-Based and Intelligent Information and Engineering Systems - KES* (2018), pp. 126 – 135.

[75] Miholca, D.-L., Czibula, G., and Czibula, I. A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. *Information Sciences 441* (2018), 152 – 170.

[76] Miholca, D. L., Czibula, G., Mircea, I. G., and Czibula, I. G. Machine learning based approaches for sex identification in bioarchaeology. In *Proceedings of the 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing - SYNASC* (2016), pp. 311–314.

[77] Miholca, D.-L., Czibula, G., and Tomescu, V. COMET: A conceptual coupling based metrics suite for software defect prediction. In *Proceedings of the 24th Knowledge-Based and Intelligent Information and Engineering Systems International Conference - KES* (2020), pp. 1 – 10. accepted for publication.

[78] Miholca, D. L., and Onicaş, A. Detecting depression from fMRI using relational association rules and artificial neural networks. In *Proceedings of the 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing - ICCP* (2017), pp. 85–92.

[79] Mitchell, T. M. *Machine Learning*, 1 ed. McGraw-Hill, Inc., 1997.

[80] Muthukumaran, K., Srinivas, S., Malapati, A., and Neti, L. B. M. Software defect prediction using augmented bayesian networks. *Advances in Intelligent Systems and Computing 614*, 1 (2018), 279–293.

[81] Nam, J., and Kim, S. Heterogeneous defect prediction. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (2015), ACM, pp. 508–519.

[82] Nath, B., Bhattacharyya, D. K., and Ghosh, A. Incremental association rule mining: A survey. *Interdisciplinary Reviews: Data Mining and Knowledge Discovery 3*, 3 (2013), 157–169.

[83] Nuñez-varela, A. S., Pérez-gonzalez, H. G., Martínez-perez, F. E., and Soubervielle-montalvo, C. Source code metrics : A systematic mapping study. *The Journal of Systems and Software 128*, Supplement C (2017), 164–197.

[84] Ogunde, A. O., Folorunso, O., and Sodiya, A. S. The design of an adaptive incremental association rule mining system. In *Proceedings of the World Congress on Engineering 2015, Volume I* (2015), pp. 1–6.

[85] Ougli, A. E., and Tidhaf, B. Optimal type-2 fuzzy adaptive control for a class of uncertain nonlinear systems using an LMI approach. *International Journal of Innovative Computing, Information and Control 11*, 3 (2015), 551–563.

[86] P. Tang M. Steinbach, A. K., and Kumar, V. *Introduction to data mining, 2nd Edition.* Addison-Wesley Longman Publishing Co., Inc., 2019.

[87] Panichella, A., Oliveto, R., and Lucia, A. D. Cross-project defect prediction models: L'union fait la force. In *Proocedings of the IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering - CSMR-WCRE* (2014), pp. 164–173.

[88] Park, M., and Hong, E. Software fault prediction model using clustering algorithms determining the number of clusters automatically. *International Journal of Software Engineering and Its Applications 8*, 7 (2014), 199–205.

[89] Perepletchikov, M., Ryan, C., Frampton, K., and Tari, Z. Coupling metrics for predicting maintainability in service-oriented designs. In *Proceedings of the Australian Software Engineering Conference - ASWEC* (2007), pp. 329–338.

[90] Perreault L., Berardinelli S., I. C., and J., S. Using classifiers for software defect detection. In *https://www.cs.montana.edu/izurieta/pubs/SEDE_2017.pdf* (2017), pp. 1–8.

[91] Poshyvanyk, D., Marcus, A., Ferenc, R., and Gyimóthy, T. Using information retrieval based coupling measures for impact analysis. *Empirical Software Engineering 14*, 1 (2009), 5–32.

[92] Radjenović, D., Heričko, M., Torkar, R., and Živkovič, A. Software fault prediction metrics: A systematic literature review. *Information and Software Technology 55*, 8 (2013), 1397–1418.

[93] Ren, J., Li, W., Wang, Y., and Zhou, L. Graph-mine: A key behavior path mining algorithm in complex software executing network. *International Journal of Innovative Computing, Information and Control 11*, 2 (2015), 541–553.

[94] Rodionova, O. Y., Oliveri, P., and Pomerantsev, A. L. Rigorous and compliant approaches to one-class classification. *Chemometrics and Intelligent Laboratory Systems 159* (2016), 89 – 96.

[95] Salam, A., and Khayal, M. S. H. Mining top-k frequent patterns without minimum support threshold. *Knowledge and Information Systems 30*, 1 (2012), 57–86.

[96] Sayyad, S., and Menzies, T. The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2015.

[97] Schwenker, F., Kestler, H. A., and Palm, G. Three learning phases for radial-basis-function networks. *Neural Networks 14*, 4-5 (2001), 439–458.

[98] Selvi, C. S. K., and Tamilarasi, A. Association rule mining with dynamic adaptive support thresholds for associative classification. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications - ICCIMA* (2007), IEEE Computer Society, pp. 76–80.

[99] Serban, G., Câmpan, A., and Czibula, I. G. A programming interface for finding relational association rules. *International Journal of Computers, Communications and Control I*, S. (2006), 439–444.

[100] Serban, G., Czibula, I. G., and Câmpan, A. Medical diagnosis prediction using relational association rules. In *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics - ICTAMI* (2008), pp. 339–352.

[101] Simon, F., Löffler, S., and Lewerentz, C. Distance based cohesion measuring. In *proceedings of the 2nd European Software Measurement Conference - FESMA* (1999), pp. 69–83.

[102] Somervuo, P., and Kohonen, T. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters 10*, 2 (1999), 151–159.

[103] Song, A., Ding, X., Li, M., Cao, W., and Pu, K. A novel binary bat algorithm for association rules mining. *ICIC Express Letters 9*, 9 (2015), 2387–2394.

[104] Stevens, W. P., Myers, G. J., and Constantine, L. L. Structured design. *IBM Systems Journal 13*, 2 (1974), 115–139.

[105] Tobergte, D. R., and Curtis, S. Data mining - practical machine learning tools and techniques. *Journal of Chemical Information and Modeling 53*, 9 (2013), 1689–1699.

[106] Van Der Maaten, L., and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605.

[107] Varade, S. Hyper-quad-tree based k-means clustering algorithm for fault prediction. *International Journal of Computer Application 76*, 5 (2013), 6–10.

[108] Wang, S., Liu, T., and Tan, L. Automatically learning semantic features for defect prediction. In *Proceedings of the 38th International Conference on Software Engineering* (2016), ACM, pp. 297–308.

[109] White, L., Jaber, K., Robinson, B., and Rajlich, V. Extended firewall for regression testing: An experience report. *Journal of Software Maintenance and Evolution 20*, 6 (2008), 419–433.

[110] Xuan, X., Lo, D., Xia, X., and Tian, Y. Evaluating defect prediction approaches using a massive set of metrics: An empirical study. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (2015), Association for Computing Machinery, p. 1644–1647.

[111] Yang, T., Asanjan, A. A., Faridzad, M., Hayatbini, N., Gao, X., and Sorooshian, S. An enhanced artificial neural network with a shuffled complex evolutionary global optimization with principal component analysis. *Information Sciences 418-419*, Supplement C (2017), 302–316.

[112] Yu, L., and Mishra, A. Experience in predicting fault-prone software modules using complexity metrics. *Quality Technology and Quantitative Management 9*, 4 (2012), 421–433.

[113] Yu-Dong, G., Sheng-Lin, L., Yong-Zhi, L., Zhao-Xia, W., and Li, Z. Large-scale dataset incremental association rules mining model and optimization algorithm. *International Journal of Database Theory and Application 9*, 4 (2016), 195–208.

[114] Zhang, C., Wu, X., Shyu, M.-L., and Peng, Q. Adaptive association rule mining for web video event classification. In *14th International Conference on Information Reuse and Integration* (2013), IEEE, pp. 618–625.

[115] Zhang, D., Tsai, J., and Boetticher, G. Improving credibility of machine learner models in software engineering. In *Advances in Machine Learning Applications in Software Engineering*. 2007, pp. 52–72.

[116] Zheng, J. Predicting software reliability with neural network ensembles. *Expert Systems with Applications 36*, 2 (2009), 2116–2122.