Universitatea Babeş–Bolyai
Facultatea de Matematică şi Informatică
str. M. Kogălniceanu nr. 1
400084, Cluj–Napoca, România
http://www.ubbcluj.ro

Refactorizarea în Modelarea Orientată Obiect

Rezumatul tezei de doctorat

Doctorand: Maria-Camelia Chisăliţă-Creţu
Conducător Ştiinţific: Prof. Dr. Militon Frenţiu

2010

DEDICAŢIE

Iubitului meu soţ Coni,

Dragostea şi sprijinul tău susţinut mi-au dat putere!

Fiului nostru David,

Timpul petrecut cu tine este realizarea noastră cea mai mare!

MULŢUMIRI

Această teză nu ar fi putut fi terminată fără ajutorul mai multor persoane faţă de care aş dori să-mi exprim sincerele mele mulţumiri.

Doresc să îmi exprim adânca apreciere faţă de coordonatorul meu, domnul profesor Militon Frenţiu, pentru încurajarea şi răbdarea neîncetată. Domnia sa mi-a dat ocazia sa lucrez într-un colectiv creativ, dându-mi libertatea de a experimenta diferite aspecte ale domeniului ingineriei softului. A avut încredere în mine şi m-a îndrumat pe parcursul elaborării tezei, crezând în capacitatea mea de a o finaliza.

Aş dori să mulţumesc părinţilor mei pentru încurajare şi înţelegere. Îi mulţumesc în mod deosebit iubitului meu soţ Coni şi fiului nostru drag David, care mi-au fost mereu aproape, încuranjându-mă şi susţinându-mă să dau ce am mai bun pe toată durata studiilor doctorale. Dragostea şi răbdarea lor continuă mi-au dat putere şi m-au motivat să finalizez teza de doctorat.

Aş dori să mulţumesc personalului Departamentului de Informatică şi tuturor colegilor mei. Domnului profesor Bazil Pârv as dori să îi mulţumesc pentru sugestiile valoroase şi discuţiile stimulatoare. Mulţumesc recenzorilor pentru că s-au oferit sa evalueze teza mea de doctorat şi pentru observaţiile pozitive pe care le-au înaintat. Doresc să îi mulţumesc în mod deosebit dragei mele colege Andreea Vescan pentru sprijinul oferit de fiecare dată când a fost nevoie, prin încurajări şi idei care mi-au impulsionat activitatea de cercetare. Multe mulţumiri colegelor mele Cristina Mihăilă şi Andreea Mihiş pentru sfaturi şi disponibilitate. De asemenea, aş dori să mulţumesc tuturor colegilor cu care am colaborat (Crina Groşan, Camelia Şerban, Anca Gog) pe toată durata realizării tezei de doctorat.

Însă, cel mai mult îi mulţumesc lui Dumnezeu pentru abilităţile cu care m-a creat şi oportunităţile care au făcut posibilă finalizarea acestei teze de doctorat.

Lista de abrevieri

Următorul tabel descrie semnificaţiile abrevierilor folosite în această teză, precizând pagina/secţiunea în care au fost definite sau prima lor apariţie din teză.

Abreviere	**Semnificaţie**	**Pagina/Secţiunea**
UoD	Universe of Discourse (Domeniul problemei)	7/2.1
AST	Abstract Syntax Tree (Arbore Abstract de Sintaxă)	10/3.1
CBO	Coupling Between Objects (Cuplarea dintre Obiecte)	14/4.4.2
RFC	Response For a Class (Răspunsul unei Clase)	14/4.4.2
DIT	Depth of Inheritance Tree (Adâncimea Arborului de Derivare)	15/4.4.2
NOM	Number Of Methods (Numărul de Metode)	14/4.4.2
NOC	Number Of Children (Numărul de Clase Derivate Direct)	14/4.4.2
$LCOM^*$	Lack Of Cohesion in Methods (Lipsa Coeziunii dintre Metode)	14/4.4.2
NOD	Number Of Descendants (Numărul de Clase Derivate)	15/4.4.2
TCC	Tight Class Cohesion (Coeziunea la Nivel de Clasă)	15/4.4.2
NOC^*	Number Of Classes (Numărul de Clase)	15/4.4.2
LOCC	Line Of Code per Class (Numărul de Linii de Cod din Clasă)	15/4.4.2
WMC	Weighted Method per Class (Ponderea Metodelor din Clasă)	15/4.4.2
SBSE	Search-Based Software Engineering (Ingineria Soft Bazată pe Căutare)	6/1.9
LANSP	Local Area Network Simulation Problem (Problema Simulării Unei Reţele Locale)	6/1.11
DSCHP	Data Structure Class Hierarchy Problem (Problema Ierarhiei de Clase cu Structuri de Date)	6/1.11
DAMP	Didactic Activity Management Problem (Problema Gestionării Activităţilor Didactice)	6/1.11
fca	forward conceptual abstraction (abstractizare conceptuală)	7/2.1
cs	conceptual specialization (specializare conceptuală)	7/2.1
GMORSP	General Multi-Objective Refactoring Selection Problem (Problema Generală Multicriterială a Selectării Refactorizărilor)	16/5.3
MORSSP	Multi-Objective Refactoring Set Selection Problem (Problema Multicriterială a Selectării Unei Mulţimi de Refactorizări)	17/5.4.1
MORSgSP	Multi-Objective Refactoring Single Selection Problem (Problema Multicriterială a Selectării Unei Refactorizări)	18/5.4.2
MORSqSP	Multi-Objective Refactoring Sequence Selection Problem (Problema Multicriterială a Selectării Unei Secvenţe de Refactorizări)	19/5.4.3
MORPBP	Multi-Objective Refactoring Plan Building Problem (Problema Multicriterială a Construirii Unui Plan de Refactorizare)	20/5.4.4
RSSGARef	Refactoring Set Selection Genetic Algorithm Refactoring-based (Algoritmul Genetic de Selectare a Mulţimii de Refactorizări, soluţie bazată pe refactorizări)	23/6.1
RSSGAEnt	Refactoring Set Selection Genetic Algorithm Entity-based (Algoritmul Genetic de Selectare a Mulţimii de Refactorizări, soluţie bazată pe entităţi)	23/6.1
RSgSGAEnt	Refactoring Single Selection Genetic Algorithm Entity-based (Algoritmul Genetic de Selectare a Unei Refactorizări, soluţie bazată pe entităţi)	28/6.7

Cuvinte cheie: Refactorizare, Ingineria softului bazată pe căutare, Optimizare multicriterială, Metrici soft.

Cuprins

Introducere

Acestă teză de doctorat este rezultatul cercetării mele în Ingineria Soft, în particular în Refactorizarea Produselor Soft, cercetare începută în anul 2002.

Produsele soft se află într-o continuă schimbare pe măsură ce sunt îmbunătăţite cu noi funcţionalităţi, dar structura lor internă se deteriorează. Refactorizarea este cunoscută ca un mijloc de îmbunătăţire a structurii interne a produselor soft orientate obiect [Fow99, MT04]. Scopul ei este acela de a opri procesul de decădere a calităţii produsului soft prin aplicarea unei serii de mici transformări care păstrează comportamentul iniţial al acestuia, fiecare dintre ele îmbunătăţind un anumit aspect al sistemului [Fow99]. Unele refactorizări sunt uşor de identificat, în timp ce, pentru altele este greu de stabilit dacă intr-adevar îmbunătăţesc structura interna a produsului soft. Este ştiut faptul că multe refactorizări utile pot îmbunătăţi un anumit aspect ala produsului soft, dar să afecteze negativ pe altele. Refactorizarea s-a integrat în mod natural în diferite metodologii de dezvoltare a produselor soft (modelul spirală [Boe88], programarea extremă [Bec99]).

Scopul acestei teze este de a investiga aplicarea unor tehnici de refactorizare adecvate în diferite situaţii. Cercetarea în modelarea conceptuală a arătat că refactorizarea se poate integra în faza de analiză a modelelor de dezvoltare a produselor soft. Pentru a gestiona diferite tipuri de variabilitate conceptuală, a fost construit un model bazat pe evoluţia biologică.

Impactul refactorizării este definit printr-un formalism nou, câteva tehnici de refactorizare particulare fiind descrise folosind notaţiile propuse. Evaluarea impactului refactorizării asupra calităţii interne a programelor poate fi folosită ca un indicator al necesităţii aplicării acesteia. Metricile soft şi modificările determinate de aplicarea refactorizărilor sunt studiate la nivel formal. Pentru a analiza modificarea valorii metricii soft în intervalul de valori al acesteia după aplicarea unei refactorizări, s-a propus o strategie de analiză în mai mulţi paşi.

Problema selectării refactorizărilor este studiată în detaliu, fiind definite formal mai multe probleme de selectare a refactorizărilor. Astfel, problemele pentru selectarea unei mulţimi de refactorizări, selectarea a unei secvenţe de refactorizări şi construirea unui plan de refactorizare sunt definite ca probleme de optimizare multicriterială cu două obiective conflictuale.

Teza se focalizează pe activitarea de selectare a refactorizărilor adecvate şi evaluarea impactului refactorizărilor asupra structurii interne a produselor soft. Conţine peste 150 referinţe bibliografice şi este împărţită în şapte capitole după cum urmează.

Primul capitol oferă o introducere în domeniul Ingineriei Soft şi poziţionează procesul de refactorizare în diferite metodologii de dezvoltare a produselor soft, prezentând o motivaţie pentru folosirea refactorizării cât şi paşii acesteia de aplicare. De asemenea, sunt prezentate aspecte generale legate de întreţinerea şi evoluţia produselor soft, precum şi câteva dintre abordările anterioare legate de formalizarea refactorizărilor. Modelarea conceptuală şi tipurile de variabilitate care apar în faza de analiză fundamentează introducerea refactorizării în această etapă de dezvoltare a produselor soft. Refactorizările utilizate în cadrul cercetării sunt prezentate informal. Abordarea evolutivă, folosită pentru rezolvarea problemelor de selectare a refactorizărilor, este descrisă în detaliu. Codul sursă pentru studiile de caz prezentate şi neajunsurile acestora care sugerează aplicarea refactorizării sunt prezentate pe pe scurt.

Capitolul 2, **Refactorizarea în Variabilitatea Modelării Conceptuale**, descrie cele trei tipuri de variabilitate conceptuală: *de construcţie*, *de abstractizare verticală* şi *de abstractizare orizontală*. Refactorizarea, abstractizarea conceptuală şi specializarea conceptuală sunt strategiile identificate pentru transformarea modelelor conceptuale. Modelul propus se bazează pe evoluţia biologică pentru a descrie transformările din cadrul modelelor studiate, prin procese *ontogenice* şi *filetice*.

Capitolul 3, **Reprezentarea Formală a Impactului Refactorizării**, formalizează impactul refactorizării asupra reprezentării structurii interne a softului prin numărul de vârfuri şi arce modificat din cadrul AST. Câteva refactorizări relevante, ca: *MoveMethod*, *MoveField*, *ExtractClass* şi *InlineClass* sunt riguros abordate şi descrise formal folosind notaţia propusă. Formalismul prezentat este aplicat pentru a studia impactul tehnicilor de refactorizare descrise asupra unui studiu de caz.

Capitolul 4, **Formalizarea Metricilor Soft. Abordare Bazată pe Impactul Refactorizării**, evidenţiază posibilitatea a conecta într-un context formal procesul de refactorizare şi procesul de evaluare a calităţii interne prin metrici soft. Se defineşte formal un număr consistent de metrici soft identificate în literatură în contextul programării orientate obiect. Se propune o tehnică de analiză a impactului refactorizării asupra calităţii interne a softului, evaluat prin metrici soft. Pentru realizarea acesteia, se enumeră obiectivele pe care dezvoltatorul le urmăreşte, o listă de categorii de impact şi o lista de reguli de evaluare aplicate pentru a obţinerea impactului final al refactorizării asupra metricii soft studiate.

Capitolul 5, **Problema Selectării Refactorizărilor. Abordare Formală a Optimizării Multicriteriale**, defineşte formal *Problema Generală de Selectare Optimă a Refactorizărilor*, descriind unele probleme multicriteriale specifice de selectare a refactorizărilor. Se definesc două obiective conflictuale: *costul refactorizării* şi *impactul refactorizării* asupra entităţilor afectate. Astfel, sunt prezentate *Problema Multicriterială de Selectare a Unei Mulţimi de Refactorizări* şi a cazul său particular, *Problema Multicriterială de Selectare a Unei Refactorizări*. Apoi, este descrisă formal *Problema Multicriterială de Selectare a Unei Secvenţe de Refactorizări*, finalizând cu *Problema Multicriterială de Construire a Unui Plan de Refactorizare*.

Capitolul 6, **Abordare Evolutivă a Problemei Selectării Refactorizărilor**, studiază abordarea evolutivă pentru *Problema Multicriterială de Selectare a Unei Mulţmii de Refactorizări*. Pentru algoritmul genetic propus, au fost studiate două reprezentări ale soluţiei: cea bazată pe refactorizări şi cea bazată pe entităţile refactorizate. Pentru *Problema Multicriterială de Selectare a Unei Refactorizări*, algoritmul genetic aplicat foloseşte o reprezentare bazată pe entităţi pentru soluţia cautată. Rezultatele obţinute sunt analizate pe baza unei strategii de refactorizare recomandate.

Capitolul 7, **Concluzii şi Teme de Cercetare**, expune concluziile principale ale abordărilor propuse. Sunt enunţate câteva potenţiale îmbunătăţiri ale cercetării desfăşurate şi aspecte specifice legate de construirea de instrumente de validare.

Contribuţiile originale introduse în această teză sunt prezentate în Capitolele 2, 3, 4, 5 şi 6. Principalele contribuţii se concentrează în jurul a trei din cei şase paşi identificaţi pentru aplicarea completă a procesului de refactorizare: alegerea refactorizării adecvate, evaluarea efectului refactorizării asupra atributelor calităţii produsului soft şi propagarea transformărilor prin refactorizare şi asupra altor etape de dezvoltare a produsului soft. Acestea sunt:

- în **variabilitatea la nivelul modelării conceptuale**:
  - O mulţime de refactorizări recomandate pentru fiecare tip de variabilitate la nivel conceptual [CC05b] (Subsecţiunea 2.1);
  - O nouă abordare în descrierea tipurilor de variabilitate conceptuală prin acţiuni specializate care sugerează transformarea între variante [CC07] (Subsecţiunea 2.1.1, 2.1.2 şi 2.1.3);
  - Un model formal bazat pe evoluţia biologică pentru a gestiona diferite tipuri de variabilitate la nivelul modelării conceptuale [CC10c] (Subsecţiunea 2.2);
- în **evaluarea şi formalizarea impactului refactorizării**:

- Un nou formalism pentru descrierea tipurilor de vârfuri şi arce din reprezentarea structurii interne prin AST [CC10b] (Subsecţiunea 3.1);
- Un nou formalism pentru descrierea impactului refactorizării folosind reprezentarea structurii interne prin AST [CC05a] (Subsecţiunea 3.1);
- Descrierea formală a diferite refactorizări folosind notaţia bazată pe impact ca: *MoveMethod*, *MoveField*, *ExtractClass* şi *InlineClass* [CC05a] (Subsecţiunea 3.2);
- Aplicarea definiţiilor pentru impactul refactorizărilor particulare asupra DAMP (Subsecţiunea 1.11.3) [CC05a] (Subsecţiunea 3.2);
- O nouă descriere formală a metricilor soft, bazată pe reprezentarea structurii interne prin AST [CC10b], pentru un număr substanţial de metrici soft definite în literatură în contextul programării orientate obiect, grupare în patru categorii (cuplare, coeziune, complexitate şi abstractizare) (Subsecţiunea 4.3);
- O nouă descriere formală a impactului refactorizărilor asupra matricilor soft [CC10b], pentru un număr substanţial de metrici soft grupate în patru categorii (cuplare, coeziune, complezitate şi abstractizare) (Subsecţiunea 4.3);
- A nouă strategie de analiză a impactului refactorizărilor asupra calităţii interne a programelor, evaluată prin metrici soft [CCcC06, CC10e] (Subsecţiunea 4.4);
- Un nou set de obiective urmărite de dezvoltator şi o listă de categorii de impact definite în contextul procesului de evaluare [CC10e] (Subsecţiunea 4.4.1);
- Un nou set de reguli de evaluare, aplicate pentru stabilirea impactului final al refactorizării asupra metricii soft studiate [CC10e] (Subsecţiunea 4.4.1);
- Validarea strategiei de analiză propuse, realizată prin două experimente [CC10e] (Subsecţiunea 4.4.2);

- în **selectarea refactorizărilor**:
  - Noi descrieri formale pentru diferite probleme de selectare a refactorizărilor definite ca probleme de optimizare multicriterială:
    - Problema Generală Multicriterială de Selectare a Refactorizărilor [CC11] (Subsecţiunea 5.3);
    - Problema Multicriterială de Selectare a Unei Mulţimi de Refactorizări [CC09c] (Subsecţiunea 5.4.1);
    - Problema Multicriterială de Selectare a Unei Refactorizări [CCV09a, CCV09b] (Subsecţiunea 5.4.2);
    - Problema Multicriterială de Selectare a Unei Secvenţe de Refactporizări [CC10f] (Subsecţiunea 5.4.3);
    - Problema Multicriterială de Construire a Planului de Refactorizare [CC10f] (Subsecţiunea 5.4.4);
  - Noi aspecte strategice pe care echipa de conducere a proiectului trebuie să le analizeze în procesul de construire al planului de refactorizare [CC10f] (Subsecţiunea 5.4.4);

- în **abordări evolutive pentru problemele MORSSP şi MORSgSP**:
  - O nouă abordare evolutivă pentru problema MORSSP, printr-un algoritm genetic cu două reprezentări ale soluţiei: cea bazată pe refactorizări [CC09c, CC09a] şi cea bazată pe entităţi [CC09b, CC09e, CC09d] (Subsecţiunile 6.2.1 şi 6.2.2);
  - O nouă abordare evolutivă pentru MORSgSP, printr-un algoritm genetic având reprezentarea soluţiei bazată pe entităţi [CC10d] (Subsecţiunea 6.7);
  - O nouă strategie de evaluare a îmbunătăţirilor sugerate de soluţia obţinută, elaborată pe baza neajunsurilor unui studiu de caz [CC10a, CC11] (Subsecţiunea 6.3.1).

1 Contextul general

Acest capitol prezintă contextul general al cercetării desfăşurate, încadrând contribuţiile personale în domeniul Ingineriei Softului şi evidenţiind procesul de refactorizare.

1.1 Întreţinerea şi evoluţia produselor soft

Un produs soft evoluează pe întreaga sa durată de viaţă. *Întreţinerea produsului soft* reprezintă unul dintre elementele esenţiale dintr-o metodologie de construire şi management a produselor soft. Fiecare funcţionalitate adăugată, eroare eliminată sau adaptarea produsului soft la noi cerinţe, îl transformă pe acesta în unul mai complex, îi schimbă proiectarea iniţială şi îi reduce calitatea internă. Întreţinerea produsului soft este parte a standardului IEEE 1219 [IEE99].

Definiţia 1.1.1([IEE99]) ***Întreţinerea softului*** *este modificarea aplicată produslui soft după instalare, cu scopul de a corecta erori, a îmbunătăţi performanţa sau alte atribute, sau pentru a adapta produsul soft la un context modificat.*

Swanson [Swa76] şi Sommerville [Som96] descriu trei tipuri de întreţinere a softului. Pornind de la clasificarea propusă de aceştia, în Standardul ISO/IEC 14764 [ISO99] întreţinerea softului este împărţită în patru categorii: corectivă, adaptivă, perfectivă şi preventivă.

Întreţinerea preventivă prin care nu alterează funcţionalitatea sistemului, este denumită şi activitate anti–regresivă (*anti–regressive work*). Termenul a fost introdus de Lehman [LR00] pentru a descrie activitatea realizată cu scopul de a scădea complexitatea unui program fără a-i schimba funcţionalitatea, aşa cum este ea percepută de către utilizatori. Activităţile anti–regresive sunt: rescrierea codului, reproiectare, refactorizare, restructurare şi redocumentare.

Astăzi, peste 80% din costurile totale ale unui produs soft sunt dedicate întreţinerii, [Pig97] şi este cunoscut faptul că întreţinerea softului cuprinde activităţi şi procese care presupun un produs soft care nu este încă instalat, ci este în plin proces de dezvoltate.

În *modelul cascadă* [Roy70], întreţinerea softului reprezintă faza finală din procesul de dezvoltare a produsului soft, după dezvoltarea completă a acestuia. Roy consideră că doar eliminarea erorilor şi ajustări minore aplicate softului trebuie realizate în faza de întreţinere de după dezvoltarea produsului soft.

Descompunerea secvenţială a activităţilor dintr-o metodologie de dezvoltare a unui produs soft, împiedică interacţiunile necesare şi răspunsul de care are nevoie echipa de dezvoltare. Acest aspect este bine evidenţiat în modelele de dezvoltare moderne ca: *modelul spirală* [Boe88] şi *programarea extremă* [Bec99].

1.2 Refactorizarea

1.2.1 Definiţii

Termentul *refactorizare* îşi are originea într-un citat al lui Deutch, care scria "proiectarea interfeţei şi factorizarea funcţională reprezintă conţinutul cheie al produsului soft şi este de departe mult mai dificil de creat sau recreat decât codul sursă" [Deu89].

Produsul soft poate fi *repetat factorizat*, adică supus unui proces de refactorizare. Constantine [CY79] a introdus termenul de *factorizare a obiectelor* în contextul orientat obiect.

Deşi a fost folosită cu mult timp înainte, prima definiţie a refactorizării este menţionată în literatură în teza de doctorat a lui Opdyke [Opd92]. Aproximativ în acelaşi timp, Chikofsky defineşte noţiunea de *restructurare* [CC90].

Definiţia 1.2.1([CC90]) ***Restructurarea*** *este o transformare dintr-o reprezentare în altă reprezentare aflată la acelaşi nivel de abstractizare, prin păstrarea comportamenului exterior al sistemului (funcţionalitate şi semantică). O astfel de transformare este de cele mai multe ori una de prezentare, cum ar fi modificarea codului sursă cu scopul de a-i îmbunătăţi structura, aşa cum este cunoscută în proiectarea structurată.*

Restructurarea permite creare de noi versiuni care implementează sau propun modificări asupra sistemului, ce nu presupun modificări legate de adăugarea unor funcţionalităţi noi. Totuşi, sistemul poate fi mai bine înţeles, ceea ce poate sugera modificări care pot îmbunătăţi anumite aspecte ale sistemului [CC90].

Definiţia 1.2.2([Opd92]) ***Refactorizările*** *sunt restructurări care păstrează comportamentul programului, care facilitează proiectarea, evoluţia şi reutilizarea aplicaţiilor orientate obiect.* ***Refactorizările*** *sunt transformări care nu afectează comportamentul programului; adică, dacă programul este folosit de două ori (înainte şi după refactorizare) cu aceleaşi date de intrare, datele de ieşire rămân aceleaşi.*

Înţelesul termenilor de restructurare şi refactorizare a evoluat de la *foarte asemănători* la *diferiţi*. S-a considerat că termenul restructurare este potrivit pentru programarea structurată, în timp ce termenul refactorizare este potrivit în programarea orientată obiect. Totuşi, termenul restructurare are un înţeles mai larg, care nu presupune păstrarea comportamentului extern al sistemului, aşa cum termenul de refactorizare îl cere. Mai târziu, Fowler [Fow04] consideră că refactorizarea este o "tehnică foarte specifică pentru a îndeplini activitatea mai generală a restructurării (...) care se regăseşte în folosirea unor mici transformări care păstrează comportamentul (denumite chiar refactorizări)".

În teza sa de doctorat, Roberts [Rob99] revizuieşte definiţia refactorizării. Aceasta acoperă atât aspectele legate de păstrarea sau nu a comportamenului programului.

Definiţia 1.2.3([Rob99]) *O* ***refactorizare*** *este un triplet ordonat $R = (pre, T, P)$ unde pre este un predicat evaluat la adevarat pentru programul R, T este o transformare a programului, iar P este o funcţie definită pe o mulţime de predicate cu valori într-o mlţime de predicate care transformă predicate pentru programe transformate de T.*

Cea mai populară definiţie a refactorizării este dată de Fowler [Fow99].

Definiţia 1.2.4([Fow99]) ***Refactorizare*** *(substantiv): o schimbare aplicată structurii interne a softului care înlesneşte înţelegerea şi face mai puţin costisitoare modificarea, fără a schimba comportamentul observabil. A* ***refactoriza*** *(verb): a restructura softul prin aplicarea unei serii de refactorizări fără modificarea comportamentului observabil.*

1.2.2 Motivaţie şi etape de aplicare

Această subsecţiune oferă o motivaţie pentru aplicarea procesului de refactorizare, aşa cum este sugerată de către diferiţi cercetători [Opd92, Fow99, Bec99, MT04]. De asemenea, este prezentat modul în care procesul de refactorizare poate fi integrat în diferite metodologii de dezvoltare a softului (*modelul spirală* [Boe88], *programarea extremă* [Bec99]). Diferite grupuri de cercetare sugerează aplicarea procesului de refactorizare în mai mulţi paşi [MT04, KIAF02], cât şi diverse clasificări ale refactorizărilor la nivelul claselor, folosind câteva criterii de organizare [Opd92, Rob99, Fow99, Ker04].

1.3 Refactorizări studiate

Această secţiune prezintă pe scurt refactorizările folosite în cadrul cercetării. Pentru a evidenţia diferite aspecte ale AST studiat, au fost alese refactorizări din diferite categorii.

1.4 Modelarea conceptuală

Modelarea conceptuală este primul pas din faza de analiză a cerinţelor şi foloseşte informaţia colectată în etapa anterioară din procesul de dezvoltare a softului [Boe88, CY79]. În contextul modelării conceptuale, *variabilitatea* [Ver04] constă în posibilitatea de construi modele conceptuale distincte dar corecte, pornind de la aceeaşi mulţime de cerinţe. Un asemenea model conceptual se numeşte *variantă*. O clasificare a tipurilor de variabilitate a fost folosită pentru a susţine ideea integrării procesului de refactorizare în modelarea conceptuală.

1.5 Formalizarea refactorizărilor prin transformarea grafelor

Această secţiune descrire reprezentarea prin grafe a structurii interne a unui program orientat obiect [MVEDJ05], reprezentare necesară pentru formalizara descrierii transformărilor aplicate prin refactorizare. Capitolul 3 introduce notaţii formale care exprimă impactul refactorizării asupra structurii interne a programelor. În Capitolul 4 se propune o abordare nouă de evaluare a efectului refactorizării asupra calităţii interne, reflectată prin metrici soft.

1.6 Atribute ale calităţii softului

Pornind de la definiţia standardului IEEE pentru *calitatea produslui soft* [IEE92] şi *caracteristicile calităţii softului* [ISO91], se poate spune că pentru definirea calităţii unui produs soft se poate defini o listă de atribute ale calităţii softului.

Cercetarea noastră se concentrează asupra atributelor calităţii produselor soft. Această secţiune prezintă pe scurt atributele calităţii interne: cuplarea scăzută, coeziunea ridicată, complexitatea şi abstractizarea adecvată datelor [Mar02].

1.7 Metrici soft studiate

Această secţiune prezintă metricile definite în contextul abordării orientate obiect, alese pentru cercetarea desfăşurată. Astfel sunt amintite: metricile de cuplare, metricile de cuplare la nivelul ierarhiei de clase, metrici de coeziune, metrici de dimensiune şi complexitate [Mar02].

1.8 Tipuri de duplicare de cod în ierarhiile de clase

Duplicarea de cod reprezintă unul din factorii care fac dificilă întreţinerea şi evoluţia softului de mari dimensiuni [Joh93, Bak95]. Clonele de cod fac parte din categoria *bad smells* pentru produsele soft [Fow99]. Această secţiune prezintă pe scurt clasificarea tipurilor de clone de cod din ierarhiile de clase [KN01], cât şi refactorizările recomandate pentru eliminarea lor.

1.9 Problema Selectării Multicriteriale a Refactorizărilor

Această secţiune descrie aspectele principale ale *Problemei Selectării Multicriteriale a Refactorizărilor*. Contextul general al Ingineriei Soft Bazată pe Căutare (Search-Based Software Engineering - SBSE), domeniul selectării refactorizărilor şi rezultate existente în literatură sunt prezentate. Conceptele cheie care definesc problema optimizării multicriteriale şi metodele de optimizare sunt discutate în această secţiune.

1.10 Abordare a soluţiei bazată pe Inteligenţa Artificială

Există mai multe abordări în cadrul SBSE legate de selectarea refactorizărilor bazate pe căutare. Această secţiune prezintă aspecte ale domeniului inteligenţei artificiale necesare rezolvării *Problemei Selectării Multicriteriale a Refactorizărilor*, definită în Capitolul 5. Alte aspecte legate de algoritmul genetic aplicat şi operatorii genetici folosiţi sunt prezentate pe scurt. Modelul evolutiv *steady–state*, folosit în cercetarea desfăşurată, precum şi metodele de normalizare a datelor, sunt prezentate în detaliu.

1.11 Studii de caz ale problemelor abordate

Trei studii de caz sunt descrise în această secţiune: Local Area Network Simulation Problem (LANSP), Data Structure Class Hierarchy Problem (DSCHP), şi un extras din Didactic Activity Management Problem (DAMP). Acestea sunt folosite pentru a evidenţia diferite aspecte legate de aplicarea refactorizărilor în cadrul cercetării desfăşurate.

2 Refactorizarea în variabilitatea modelării conceptuale

Acest capitol studiază aplicarea refactorizării în contextul variabilităţii la nivel modelării conceptuale şi descrie diferite tipuri de modificări care presupun aplicarea unor refactorizări specifice. De asemenea, se propune un model bazat pe evoluţia biologică pentru descrierea transformărilor modelelor conceptuale.

2.1 Tipuri de variabilitate în modelarea conceptuală

În contextul modelării conceptuale, *variabilitatea* [Ver04] constă în posibilitatea de a construi modele conceptuale diferite dar corecte, pornind de la aceleaşi cerinţe. Un asemenea model se numeşte *variantă*.

Variabilitatea în modelarea conceptuală a arătat că refactorizarea este o tehnică utilă în migrarea de la un model conceptual la altul. Pentru fiecare tip de variabilitate studiat s-au identificat posibile transformări: refactorizare (r), specializare conceptuală (conceptual specialization - cs) şi abstractizare conceptuală (forward conceptual abstraction - fca).

2.1.1 Variabilitatea de construcţie

Definiţia 2.1.1([Ver04]) ***Variabilitatea de construcţie*** *reprezintă posibilitatea de a modela concepte în domeniul problemei (Universe of Discourse - UoD) folosind construcţii diferite în acelaşi limbaj de modelare.*

Refactorizări Recomandate

În variabilitatea de construcţie, conceptele din UoD au acelaşi înţeles în toate variantele. Ele pot fi reprezentate de clase (entităţi), atribute sau relaţii. Există câteva tipuri de refactorizări care se pot aplica, corespunzător tipurilor de variabilitate de construcţie studiate:

- folosirea unei clase (entităţi) sau a unei mulţimi de atribute (proprietăţi);
- folosirea unei metode sau a unui atribut;
- folosirea claselor derivate sau a constantelor.

Efortul necesar transformării variantelor constă în aplicarea unui număr mic de refactorizări cu complexitate redusă. Acest tip de variabilitate este utlizat la trecerea din faza de analiză la etapa de proiectare a produsului soft. Astfel, se poate spune că *variabilitatea de construcţie* este folosită deja în modelarea conceptuală. Alte rezultate obţinute în refactorizarea modelelor conceptuale arată că este puţin probabil să apară obstacole în transformarea modelelor conceptuale folosind acest tip de variabilitate [DDN00, DB04, DBM03].

2.1.2 Variabilitate de abstractizare verticală

Definiţia 2.1.2([Ver04]) ***Variabilitatea de abstractizare verticală*** *referă posibilitatea de a modela concepte în UoD într-o manieră mai mult sau mai puţin generică (abstractă).*

Refactorizări Recomandate

Există două direcţii de navigare a variabilităţii de abstractizare verticală. Prima se referă la posibilitatea de migra de la un model general la unul mai concret, iar a doua constă în creşterea nivelului de abstractizare prin eliminarea aspectelor concrete sau adăugarea unor parametri. În [CC05a] sunt descrise categoriile de refactorizări necesare pentru a migrra între asemenea modele conceptuale. S-au propus soluţii adecvate pentru cele două tipuri de transformare care apar în variabilitatea de abstractizare verticală:

- migrarea către o variantă mai abstractă (generică);
- migrarea către o variantă mai concretă (particulară).

Abstractizarea conceptuală este necesară pentru a identifica şi implementa noi concepte şi comportament specializat care să transforme modele conceptuale particulare în altele mai abstracte. Tehnicile de refactorizare au o utilitate limitată în această direcţie de navigare a variabilitaţii. Noile variante au avantajul adaptabilităţii şi flexibilităţii crescute.

Acest tip de a variabilitate este întâlnit în procesul de simplificare a proiectării unui produs soft existent [Par79, GP95, FS97, DDN00]. Modificările asupra unui sistem se fac mult mai uşor dacă modelul conceptual este mai abstract şi mult mai greu dacă modelul conceptual este mai concret.

2.1.3 Variabilitate de abstractizare orizontală

Definiţia 2.1.3([Ver04]) ***Variabilitatea de abstractizare orizontală*** *referă posibilitatea de modelare a conceptelor din UoD folosind proprietăţi diferite.*

Refactorizări Recomandate

Migrarea între variantele *cmA* şi *cmB* se realizează folosind varianta *cmC* ca şi model conceptual intermediar. Pentru a realiza acest lucru, este necesară aplicarea următorilor paşi:

- *Pasul 1:* stabilirea unei relaţii de echivalenţă între cele două dimensiuni, prin transformarea variantei *cmA* în varianta *cmC*:
  1. *specializarea prin adăugarea de clase noi*;
  2. *redistribuirea responsabilitţilor*;
  3. *generalizarea prin adăugarea unei clase de bază*;
- *Pasul 2:* păstrarea unei dimensiuni primare, prin transformarea variantei *cmC* în varianta *cmB*:
  1. *redistribuirea responsabilităţilor*;
  2. *eliminarea claselor specializate.*

În variabilitatea de abstractizare orizontală, conceptele din UoD sunt modelate folosind definiţii semantice diferite, descrierea variantelor bazându-se pe proprietăţi diferite.

Proprietăţile pot fi sau nu *vizibile* şi *izolate*. Acestea pot fi clasificate în *proprietăţi ale dimensiunii primare* (care pot fi vizibile şi izolate de altele) şi *proprietăţi ale dimensiunii secundare* (care nu sunt vizibile şi nu pot fi izolate de altele) [Ver04].

2.2 Model de evoluţie în variabilitatea de modelare conceptuală

La nivelul modelării conceptuale, variabilitatea evidenţiază un proces de evoluţie între variantele unui tip de variabilitate specific. Acest proces este similar cu procesul de evoluţie biologică prezentat de Maturana and Varela în [MV98].

Beneficiarii unui produs soft pot solicita integrarea unor funcţionalităţi noi, ceea ce presupune modificări care sunt interpretate ca perturbări ale evoluţiei produsului soft. Modificările sunt introduse în sistem prin refactorizare, abstractizare conceptuală şi specializare conceptuală. Există două tipuri de evoluţie în biologie: *filogenia* and *ontogenia* [MV98]. Primul tip de evoluţie sugerează evoluţia ca specie, iar cea de-a doua indică evoluţia individuală a fiinţelor. Pentru tipurile de variabilitate studiate, s-a dezvoltat un model de evoluţie.

2.2.1 Variabilitatea de modelare conceptuală ca evoluţie biologică

Evoluţia în Variabilitatea de Construcţie

Evoluţia ontogenică a modelelor conceptuale care sunt perturbate de mici transformări nu afectează fundamental modelul dezvoltat. Aceste transformări corespund schimbărilor dintre *atribute şi entităţi* sau *atribute şi metode*. *Definirea de constante* în variabilitatea de

construcţie indică un proces de evoluţie filetică prin adăugarea (abstractizare conceptuală) sau eliminarea (specializarea conceptuală) a constantelor din modelul conceptual.

Evoluţia în Variabilitatea de Abstractizare Verticală

Reducerea nivelului de abstractizare a unui model conceptual înseamnă eliminarea informaţiei inutile pentru o variantă concretă. Procesul de simplificare constă în aplicarea refactorizărilor care elimină informaţiile irelevante în modelul final. *Creşterea nivelului de abstractizare* presupune adăugarea de informaţie suplimentară, obţinută prin abstractizare conceptuală.

Evoluţia în Variabilitatea de Abstractizare Orizontală

Evoluţia filetică apare în primul pas de migrare în cadrul acestui tip de variabilitate, prin *adăugarea unui nivel de vizibilitate nou* modelului conceptual, ceea ce creşte nivelul de complexitate prin abstractizare conceptuală. Pentru a *reduce numărul de dimensiuni vizibile*, refactorizarea se aplică în specializarea conceptuală.

2.2.2 Abordare formală

Această secţiune prezintă elementele de bază necesare formalizării variabilităţii la nivel conceptual, ca evoluţie ontogenică sau filetică. Astfel, se definesc formal noţiuni ca: *model conceptual*, *refactorizare*, *abstractizare conceptuală* şi *specializare conceptuală*.

Pornind de la noţiunile introduce anterior, se descriu formal procesele *ontogenice* şi *filetice*. Cele trei tipuri de variabilitate în modelarea conceptuală se formalizează ca procese biologice.

S-au definit formal şi relaţiile existente între diferite modele conceptuale aflate la acelaşi nivel de abstractizare a procesului de dezvoltare. Astfel, s-au definit formal *echivalenţa ontogenică* şi *dominanţa filetică*. Noţiunile definite sunt prezentate în Tabelul 1.

Tip de evoluţie	Realizată prin	Tip de variabiltate în modelarea conceptuală		
		Construcţie	Abstractizare Verticală	Abstractizare orizontală
ontogenică	r	E – A A – M	-	-
filetică	fca	Coduri Multiple	Specific la Genetic	Adăugare dimensiune primară
	cs		Generic la Specific	Eliminare dimensiune secundară

Tabela 1: Variabilitatea în modelarea conceptuală definită ca evoluţie *ontogenică* şi *filetică*

2.3 Concluzii şi direcţii de cercetare

Variabilitatea apare în aproape orice proces de modelare, iar utilizarea ei ajută analiştii să migreze între decizii şi să valideze echivalenţa modelelor. Refactorizarea, aplicată în proiectare şi implementare, îşi extinde aplicabilitatea la nivelul analizei. Rezultatele originale prezentate în acest capitol au fost publicate în articolele [CC05b, CC07, CC10c].

3 Reprezentarea formală a impactului refactorizării

Impactul refactorizării asupra reprezentării structurii interne este exprimat ca număr de vârfuri şi arce afectate în AST. Este prezentată o nouă abordare formală a descrierii refactorizărilor.

3.1 Reprezentarea formală a impactului refactorizării asupra structurii interne a codului sursă

Această secţiune introduce nouţiuni noi în reprezentarea structurii interne reflectată ca modificări asupra AST. Pornind de la notaţiile prezentate în [DBM03], sunt necesare notaţii suplimentare pentru a formaliza impactul refactorizării asupra vârfurilor şi arcelor din AST.

3.1.1 Formalisme ale AST bazate pe vârfuri şi arce

Definiţia 3.1.1([CC10b]) *Fie $\Sigma_c = \{M, A, P, L, E\}$ mulţimea tipurilor de vârfuri faţă de care vârful c de tip Class poate fi legat prin diferite tipuri de arce şi $T(c)$ AST corespunzător. Atunci,* ***numărul total de vârfuri*** *de tip $X, X \in \Sigma_c$, notat prin $\#X(T(c))$, este numărul total de vârfuri tip X din $T(c)$.*
Definiţia 3.1.2([CC10b]) *Fie $\Gamma_{cls} = \{i, t, c, a, u\}$ mulţimea tipurilor de arce prin care vârful cls de tip Class este legat cu alte vârfuri, ϵ o expresie regulară peste Γ_{cls} şi $T(cls)$ AST corespunzător. Atunci,* ***numărul total de arce*** *de tip $\gamma, \gamma \in \Gamma_{cls}$, notat prin $\#ET(cls, \gamma)$, este numărul de arce de tip $\gamma, \gamma \in \Gamma_{cls}$ sau expresiile regulare ϵ peste Γ_{cls} care conţin literalul γ, incident la $T(cls)$.*

3.1.2 Formalisme bazate pe vârfuri şi arce pentru impactul refactorizării

Definiţia 3.1.3([CC10b]) *Fie $\Sigma_c = \{M, A, P, L, E\}$ mulţimea de vârfuri faţă de care vârful c de tip Class poate fi legat prin diferite tipuri de arce, $T(c)$ AST corespunzător, r o refactorizare aplicată asupra $T(c)$, $X(T(c), r+)$ mulţimea vârfurilor adăugate lui $T(c)$ după aplicarea refactorizării r şi $X(T(c), r-)$ mulţimea de vârfuri eliminate din $T(c)$ după aplicarea refactorizării r. Atunci, pentru un vârf de tip $X, X \in \Sigma_c$ se definesc:*

i. ***numărul de vârfuri de tip X adăugat prin aplicarea refactorizării*** *r, notat prin $\#X(T(c), r+)$, este numărul de vârfuri de tip X adăugate la $T(c)$, după aplicarea refactorizării r;*
ii. ***numărul de vârfuri de tip X eliminat prin aplicarea refactorizării*** *r, notat prin $\#X(T(c), r-)$, este numărul de vârfuri de tip X eliminate din $T(c)$, după aplicarea refactorizării r.*

Definiţia 3.1.4([CC10b]) *Fie $\Gamma_{cls} = \{i, t, c, a, u\}$ mulţimea de arce prin care vârful cls de tip Class poate fi legat de alte vârfuri, $T(cls)$ AST corespunzător, r o refactorizare aplicată asupra lui $T(cls)$, $ET(cls, \gamma, r+)$ mulţimea de arce de tip γ adăugate lui $T(cls)$ după aplicarea refactorizării r şi $ET(cls, \gamma, r-)$ mulţimea de arce de tip γ eliminate din $T(cls)$ după aplicarea refactorizării r, $\gamma \in T(cls)$. Atunci, pentru tipul de arc $\gamma, \gamma \in \Gamma_{cls}$, se definesc:*

i. ***numărul de arce de tip γ adăugate după aplicarea refactorizării*** *r, notat prin $\#ET(cls, \gamma, r+)$, este numărul de arce de tip γ adăugate la $T(cls)$, după aplicarea refactorizării r;*
ii. ***numărul de arce de tip γ eliminate după aplicarea refactorizării*** *r, notat prin $\#ET(cls, \gamma, r-)$, este numărul de arce de tip γ eliminate din $T(cls)$, după aplicarea refactorizării r.*

3.2 Descrierea refactorizărilor bazată pe impact

Pentru descrierea formală a refactorizărilor, a fost studiată o abordare bazată pe impact. Tehnicile de refactorizare studiate sunt: *MoveMethod*, *MoveField*, *ExtractClass* şi *InlineClass*. S-au folosit notaţiile introduse în [DBM03] şi descrierea impactului refactorizării prin numărul de vârfuri şi arce ale AST afectate după aplicarea refactorizării. Există câteva aspecte legate de impactul refactorizărilor care nu se pot descrie folosind AST.

Experiment. Codul sursă folosit pentru acest studiu este descris în Subsecţiunea 1.11.1, pentru care s-au aplicat refactorizările *MoveMethod*, *MoveField*, *ExtractClass* şi *InlineClass*. Pentru fiecare refactorizare s-a descris formal impactul asupra AST, folosind notaţiile prezentate în Secţiunea 1.5. Formulele identificate sunt aplicate pentru fiecare clasă a codului sursă, *înainte* şi *după* aplicarea fiecărei refactorizări.

Subsecţiunea 3.2.1 prezintă refactorizarea *MoveMethod*. Folosind formalismul propus pentru descrierea impactului refactorizării asupra reprezentării structurii interne, se descrie formal impactul asupra clasei sursă A şi clasei destinaţie B. În Subsecţiunea 3.2.2 se descrie formal impactul refactorizării *MoveField* prin care atributul `attr` este mutat din clasa sursă `A` în clasa destinaţie `B`. Impactul refactorizărilor *ExtractClass* şi *InlineClass* este definit formal în Subsecţiunile 3.2.3 şi 3.2.4.

3.3 Concluzii şi direcţii de cercetare viitoare

Pentru descrierea formală a refactorizărilor, a fost studiată o abordare bazată pe impact. S-au folosit notaţiile introduse în [DBM03] pentru descrierea structurii interne prin AST.

Capitolul prezintă impactul refactorizărilor asupra structurii interne reflectată prin modificări asupra AST. S-au introdus notaţii formale necesare pentru a evidenţia numărul de vârfuri şi arce de tipuri specifice şi pentru evaluarea impactului refactorizărilor asupra AST, folosind lucrările [CC05a, CC10b].

Direcţiile viitoare de cercetare pot include următoarele aspecte: un catalog de descrieri formale ale impactului refactorizărilor asupra reprezentării structurii interne ca AST pentru alte refactorizări relevante din diferite categorii şi o analiză aprofundată a refactorizărilor compuse, unde unele modificări asupra structurii interne pot fi anulate de aplicarea ulterioară a altor refactorizări.

4 Formalizarea metricilor soft. Abordare bazată pe impactul refactorizării

Metricile soft reprezintă un instrument esenţial în anumite domenii ale Ingineriei Softului. Acestea sunt folosite pentru a evalua calitatea şi complexitatea sistemelor soft, dar şi pentru a înţelege şi furniza indicii legate de aspecte sensibile ale produslui soft. Există multe metrici soft definite informal în literatură. Pentru o parte din acestea am oferit o descriere formală, doar câteva dintre ele fiind folosite ulterior. Capitolul conţine o descriere formală a metricilor soft şi a modificărilor corespunzătoare determinate de aplicarea refactorizărilor. De asemenea, s-a propus o strategie de analiză în mai mulţi paşi pentru evidenţierea modificării valorilor metricilor soft în intervalul corespunzător, după aplicarea refactorizării.

4.1 Utilizarea metricilor soft în procesul de refactorizare

În general, metricile soft sunt aplicate pentru evaluarea factorilor de calitate internă [FP97]. De aceea, evaluarea impactului refactorizării asupra calităţii interne a programelor poate folosită ca un indicator al necesităţii de transformare prin refactorizare. Figura 1 prezintă existenţa unei duble conexiuni între procesul de refactorizare şi procesul de evaluare a calităţii interne prin metrici soft, într-un context formal (AST).

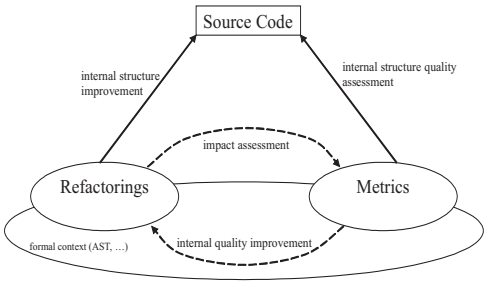

Figura 1: Conexiunea între *procesul de refactorizare* şi procesul de evaluare (*metrici soft*)

Mens şi Tourwe [MT04] sugerează existenţa a şase paşi în aplicarea procesului de refactorizare, metricile soft fiind folosite *după* refactorizarea propriu-zisă, ca un mijloc de evaluare a calităţii produsului soft. Folosind abordarea propusă de Kataoka et al. [KIAF02], evaluarea impactului refactorizării ar trebui realizată *înainte* de aplicarea propriu-zisă a refactorizării. Astfel, metricile soft au relevanţă atât înainte cât şi după aplicarea procesului de refactorizare.

Existenţa unei duble conexiuni între refactorizare şi metrici soft cât şi formalismele pentru reprezentarea structurii interne evidenţiază existenţa unor formalisme sugestive pentru descrierea metriclor soft.

4.2 Formalizarea metricilor soft orientate obiect

Formalismul introdus pentru descrierea numărului de vârfuri şi arce din descrierea structurii interne reprezentată ca AST (vezi Secţiunea 3.1) este necesar pentru a descrie formal metricile soft orientate obiect.

Metricile soft definite în literatură în contextul abordării orientate obiect au fost grupate în patru categorii: cuplare, coeziune, complexitate şi abstractizare, prezentate în [Mar02]. Aceste metrici sunt definite formal folosind notaţiile introduse în Secţiunea 3.1.

Folosind descrierea informală pentru metricile de cuplare [Mar02] şi notaţia formală din [CC10b], Subsecţiunea 4.2.1 formalizează metricile de cuplare a accesului, a metodelor, a serviciilor, la nivelul ierarhiilor de clase şi sistemului.

Cercetarea în coeziunea orientată obiect a început cu evidenţierea lipsei acesteia, ulterior fiind abordate coeziunea strânsă şi slabă, directă şi indirectă. Subsecţiunea 4.2.2 descrie formal metricile de coeziune prezentate în [Mar02].

Metricile pentru măsurarea complexităţii structurale la nivelul claselor şi al sistemului sunt studiate formal în Subsecţiunea 4.2.3.

4.3 Descrierea formală a impactului refactorizării asupra metricilor soft orientate obiect

În [DBM03] se sugerează că metricile orientate obiecte pentru evaluarea calităţii interne pot fi descrise în termenii AST extins. Folosind notaţia din [CC05a], formalizarea impactului refactorizării asupra structurii interne (vezi Secţiunea 3.1) este translatată în impactul refactorizării asupra metricilor de evaluare a calităţii interne. În [ML02] se introduce un formalism bazat pe grafe pentru descrierea metricilor soft, folosit de asemenea şi în [CC05a].

Descrierea formală a refactorizărilor şi definirea formală a unei mulţimi consistente de metrici soft orientate obiect (prezentate în [Mar02]) furnizează informaţii cu privire la impactul aplicării unei refactorizări spicifice asupra oricărei metrici de calitate internă, înainte de a o aplica propriu-zis. Descrirea termenilor definiţiei metricii soft în termenii AST şi descrierea formală a refactorizărilor sunt proiectate asupra unor metrici particulare.

Folosind metricile soft de cuplare [Mar02] descrise formal în Subsecţiunea 4.2.1 şi notaţia din [CC10b], impactul refactorizării asupra cuplării de acces, de methodă, de servicii, la nivelul ierarhiei de clase şi sistem sunt definite ca şi consecinţe ale procesului de refactorizare în Subsecţiunea 4.3.1.

Folosind formalizarea metricilor de coeziune [Mar02], introduse în Subsecţiunea 4.2.2 şi notaţia din [CC10b], Subsecţiunea 4.3.2 defineşte impactul refactorizării asupra metricilor de coeziune ca şi consecinţe ale procesului de refactorizare.

Pentru metricile de dimensiune şi complexitate descrise formal în Subsecţiunea 4.2.3 şi aplicând notaţia din [CC10b], impactul refactorizării asupra dimeniunii sistemului, a clasei şi asupra complexităţii structurale este definit ca şi consecinţe ale procesului de refactorizare în Subsecţiunea 4.3.3.

4.4 Analiza impactului refactorizării asupra metricilor soft

Definiţiile din Secţiunea 3.1 sugerează posibilitatea de a descrie formal modificările asupra AST după refactorizare. Prin translatarea modificărilor asupra structurii programului în modificări asupra unor metrici soft, se obţine impactul refactorizării asupra metricilor soft de calitate internă.

Analog abordării prezentate în [DBM03, TK03], în [CCcC06, CC10e] se propune o mulţime de tipuri de impact asupra metricilor soft. Astfel, calculul valorii unei metrici soft este clasificat în una din trei categorii: ***creşte*** ($\triangle$), ***scade*** ($\triangledown$) sau ***irelevant*** ($\bigcirc$).

Definiţia formală a fiecărei metrici soft (vezi Secţiunea 4.2) este descompusă în unul sau mai mulţi termeni. Impactul imediat al aplicării refactorizării asupra acestor termeni este descris ca: ***nul*** (0), ***pozitiv*** (+) sau ***negativ*** (−).

Combinarea celor două tipuri de impact (*intervalul de valori al metricii soft* şi *direcţia de modificare a valorii*) produce impactul final exprimat în Tabela 2. Astfel, există trei tipuri de categorii de impact cu trei tipuri de intervale de valori pentru metrici specifice.

4.4.1 Etape de evaluare a impactului

Analiza impactului presupune execuţia a cinci paşi. Categoria de impact a refactorizării identificată pentru fiecare termen (*nul*, *pozitiv*, *negativ*) şi regulile de evaluare sunt folosite

Categorie de impact	**Impactul asupra metricii soft, Valoare în interval**		
	nul, 0	*pozitiv*, $(0, \infty)$	*negativ*, $(-\infty, 0)$
$\triangle$	$\triangle_\bullet$ *valoare cautata mare, dar nu este afectata* (***fara impact***)	$\triangle_+$ *valoare cautata mare, si creste* (***impact pozitiv***)	$\triangle_-$ *valoare cautata mare, si descreste* (***impact negativ***)
$\triangledown$	$\triangledown_\bullet$ *valoare cautata mica, dar nu este afectata* (***fara impact***)	$\triangledown_+$ *valoare cautata mica, si creste* (***impact negativ***)	$\triangledown_-$ *valoare cautata mica, si descreste* (***impact pozitiv***)
$\bigcirc$	$\odot$ ***fara impact***	$\oplus$ ***impact pozitiv***	$\ominus$ ***impact negativ***

Tabela 2: Categorii de impact asupra metricilor soft

pentru a determina impactul final al refactorizării asupra metricii soft, aşa cum este descris de Tabela 2. Regulile de evaluare sunt descrise în [CC10e].

Analiza impactului constă în aplicarea următorilor paşi:

1. se alege codul sursă pentru care se va studia impactul refactorizării asupra structurii interne prin metrici soft, notat prin SC;
2. se alege o mulţime sugestivă de refactorizări care îmbunătăţeşte structura internă, notată prin $SR = \{r_1, r_2, \ldots, r_n\}$;
3. se alege o mulţime relevantă de metrici soft care să evalueze atributele interne ale calităţii, ca $SM = \{sm_1, sm_2, \ldots, sm_m\}$, unde $\forall sm_i, sm_i \in SM$; se defineşte o mulţime de termeni $sm_i = \{t_{i_1}, t_{i_2}, \ldots, t_{i_k}\}, k \in \mathcal{N}$, care vor fi folosiţi pentru a evalua termenii $sm_i, i = \overline{1,m}$;
4. se construieşte o tabelă de impact, cu n linii şi *k+1* coloane, pentru fiecare metrică soft $sm_i, i = \overline{1,m}$, cu următoarele elemente:
   (a) fiecare linie din tabelă conţine o refactorizare r_i care se va aplica codului sursă SC, unde $r_i \in SR,\ i = \overline{1,n}$;
   (b) fiecare coloană din tabelă conţine termenul $t_{i_j}, j = \overline{1,k}$, a metricii soft studiate $sm_i, i = \overline{1,m}$, ce se va completa cu impactul imediat al refactorizării r_i;
   (c) valoarea ultimei coloane din tabelă ($col_{i_{k+1}}$) se va calcula pe baza regulilor de evaluare;
5. se construieşte o tabelă de impact a refactorizărilor, cu n linii şi m coloane, completată cu valorile din coloana $col_{i_{k+1}}$ corespunzătoare fiecărei metrici soft studiate din mulţimea SM care reflectă impactul final al fiecărei refactorizări aplicate din mulţimea SR.

4.4.2 Validarea analizei bazată pe impact

Pe baza descrierii formale a metricilor soft din Subsecţiunea 4.4.1, se poate realiza validarea analizei bazată pe impact. Scopul este demonstrarea faptului că impactul asupra metricilor soft sugerat de abordarea curentă este similar celui indicat de definiţia formală. Pentru realizarea validării, s-au derulat două experimente prezentate în [CC10e].

Experimentul 1. Scopul este demonstrarea faptului că pentru o mulţime de refactorizări şi o mulţime de metrici soft, impactul refactorizării asupra metricilor soft, ce a fost calculat pe baza abordării propuse, este similar rezultatelor corespunzătoarele impactului refactorizărilor din definiţia formală a metricilor soft. Pentru patru tehnici de refactorizare definite formal în Secţiunea 3.1 (*MoveMethod, MoveField, ExtractClass* şi *InlineClass*), a fost aleasă o mulţime de metrici soft relevante: *NOM, NOC, CBO, RFC* şi $LCOM^*$, formalizate în Secţiunea 4.2. Tabelele E.1, E.2, E.3, E.4 şi E.5 din **Anexa E** prezintă impactul refactorizării asupra

metricilor soft propuse pentru clasa sursă A şi clasa destinaţie B, pentru o anumită direcţie de modificare a valorii metricii soft din intervalul de valori corespunzătoare ($\triangle$, $\triangledown$ sau $\bigcirc$).

Experimentul 2. Acesta validează abordarea propusă prin studierea unei mulţimi de refactorizări şi a impactului acestora asupra unei mulţimi de metrici soft pentru Problema DSCH (vezi Subsecţiunea 1.11.2). Situaţiile care necesită refactorizare sunt indicate de duplicările de cod din ierarhia de clase (vezi Secţiunea 1.8). În [CCcC06] validarea pentru abordarea propusă a fost realizată pentru eliminarea problemei redundanţei codului sursă. Pentru Problema DSCH au fost identificate şi eliminate şase tipuri de duplicări de cod (vezi Secţiunea 1.8). Metricile soft studiate investighează câteva aspecte ale calităţii interne ca: ponderea ierarhiei de clase (prin metrica NOC^* aplicată sistemului), dimensiunea şi complexitatea claselor (prin metricile LOCC, NOM şi WMC), cuplarea (prin metricile CBO şi RFC), coeziunea (TCC) şi cuplarea la nivelul ierarhiei de clase (prin metricile DIT, NOC şi NOD). Evaluarea calităţii interne a programului prin metrici soft a fost realizată înainte şi după eliminarea redundanţei de cod sursă. Tabelele F.1, F.2, F.3, F.4, F.5 şi F.6 din **Anexa F** prezintă impactul refactorizării asupra metricilor soft asupra claselor afectate.

4.4.3 Limitările abordării propuse

Pentru câteva metrici soft studiate nu este disponibilă o definiţie formală a impactului refactorizării (CC, SIZE1, SIZE2). Astfel, modelul de reprezentare a structurii interne a programului se poate extinde. Principalul avantaj este reprezentat de uşurinţa în descrierea formală a diferitelor aspecte legate de controlul fluxului şi complexitate, cu neajunsul dat de utilizarea unui model mai greu de gestionat.

O altă limitare este reprezentată de evaluarea dificilă a termenilor din definiţia formală a metricilor soft care au impact imediat diferit, adică o parte impact negativ ($-$) şi o altă parte impact pozitiv ($+$). Acete situaţii necesită o analiză complexă care înseamnă evaluarea semantică a AST pentru codul sursă analizat, dacă adăugarea sau eliminarea anumitor tipuri de arce are impact asupra termenilor din definiţia formală. Acesta este un aspect important al abordării propuse, definiţia formală bazându-se pe mulţimi de tipuri de vârfuri şi arce.

4.5 Concluzii şi direcţii de cercetare viitoare

Evaluarea impactului refactorizării asupra calităţii interne a programului poate fi folosită ca un indicator al necesităţii de transformare prin refactorizare. Descrirea formală refactorizărilor (vezi Capitolul 3) şi definirea formală a unei mulţimi reprezentative de metrici soft de evaluare a calităţii programelor orientate obiect în [CC10b] indică utilitatea impactului în aplicarea unor refactorizări specifice asupra unor metrici soft, înainte de a aplica refactorizarea propriu-zisă.

Pentru realizarea analizei de impact a refactorizării asupra metricilor soft, s-a propus o strategie în cinci paşi. Aceasta permite stabilirea modificării valorii metricii soft în intervalul de valori, după aplicarea uneia sau mai multor refactorizări.

Capitolul se bazează pe câteva articole publicate [CCcC06, CC10b, CC10e]. Aspectele care trebuie analizate în viitor sunt: îmbunătăţirea modelului de reprezentare a structurii interne a programelor prin includerea informaţiilor legate de controlul fluxului, descrierea completă a caracteristicilor claselor, descrierea formală a duplicărilor de cod din cadrul ierarhiilor de clase; automatizarea procesului de analiză a impactului refactorizărilor, esenţial în gestionarea unui volum mare de combinaţii de refactorizări şi metrici soft.

5 Problema selectării refactorizărilor. Abordare formală a optimizării multicriteriale

Acest capitol defineşte formal câteva variante ale *Problemei Selectării Refactorizărilor*. Aceasta este abordată ca o problemă de optimizare multicriterială, iar Capitolul 6 propune o soluţie evolutivă, unde obiectivele de optimizat sunt agregate într-o singură funcţie de optimizat.

5.1 Context general

Această secţiune prezintă contextul general al cercetării în selectarea refactorizărilor, precum şi motivaţia abordării problemei şi un scenariu de lucru.

5.2 Rezultate ale cercetărilor înrudite

Această secţiune prezintă rezultatele unor cercetări asociate problemei studiate.

5.3 Definiţia formală a Problemei Multicriteriale de Selectare a Refactorizărilor

Pentru a enunţa *Problema Generală Multicriterială de Selectare a Refactorizărilor* (*General Multi-Objective Refactoring Selection Problem, GMORSP*) au fost introduse câteva caracteristici. $SE = \{e_1, \ldots, e_m\}$ este mulţimea de entităţi soft, adică clase, atribute, metode ale claselor, parametri formali ai metodelor sau variabile locale declarate în implementarea metodelor. Acestea sunt considerate componenete de bază în programarea orientată obiect, legate prin relaţii de dependenţă.

Un sistem soft SS constă într-o mulţime de entităţi soft SE legate prin diferite tipuri de dependenţe, definite ca:
$SED = \{\mathbf{u}ses\mathbf{A}ttribute, \mathbf{c}alls\mathbf{M}ethod, \mathbf{s}uper\mathbf{C}lass, \mathbf{a}ssociatedwith\mathbf{C}lass, \mathbf{n}o\mathbf{D}ependency\}$,
$ed : SE \times SE \rightarrow SED$,

$$ed(e_i, e_j) = \begin{cases} \text{uA}, & \text{dacă } \textit{metoda } e_i \textit{ foloseşte atributul } e_j \\ \text{cM}, & \text{dacă } \textit{metoda } e_i \textit{ apelează metoda } e_j \\ \text{sC}, & \text{dacă } \textit{clasa } e_i \text{ este } \textit{clasă de bază directă} \text{ pentru } \textit{clasa } e_j \\ \text{aC}, & \text{dacă } \textit{clasa } e_i \text{ este } \textit{asociată clasei } e_j \\ \text{nD}, & \text{altfel} \end{cases}, \quad (1)$$

unde $1 \leq i, j \leq m$.

O mulţime de refactorizări relevante aplicate diferitelor tipuri de entităţi soft din SE este $SR = \{r_1, \ldots, r_t\}$. Refactorizări specifice pot fi aplicate unor tipuri particlare de entităţi soft, de exemplu refactorizarea *RenameMethod* se poate aplica doar metodelor, iar refactorizarea *ExtractClass* se poate aplica doar claselor. Astfel, o funcţie care stabileşte aplicabilitatea pentru o mulţime de refactorizări alese SR asupra unei mulţimi de entităţi soft SE, definită ca: $ra : SR \times SE \rightarrow \{\mathbf{T}rue, \mathbf{F}alse\}$,

$$ra(r_l, e_i) = \begin{cases} \text{T}, & \text{dacă } r_l \text{ se poate aplica entităţii } e_i \\ \text{F}, & \text{altfel} \end{cases}, \quad (2)$$

unde $1 \leq l \leq t$, $1 \leq i \leq m$.

Dependenţele existente între refactorizări când acestea sunt aplicate aceleaşi entităţi soft sunt evidenţiate de următoarea funcţie:
$SRD = \{\mathbf{B}efore, \mathbf{A}fter, \mathbf{A}lways\mathbf{B}efore, \mathbf{A}lways\mathbf{A}fter, \mathbf{N}ever, \mathbf{W}henever\}$,

$rd : SR \times SR \times SE \to SRD$,

$$rd(r_h, r_l, e_i) = \begin{cases} \text{B}, & \text{dacă } r_h \text{ se poate aplica entităţii } e_i \text{ doar înaintea aplicării lui } r_l,\ r_h < r_l \\ \text{A}, & \text{dacă } r_h \text{ se poate aplica entităţii } e_i \text{ doar după aplicarea lui } r_l,\ r_h > r_l \\ \text{AB}, & \text{dacă } r_h \text{ şi } r_l \text{ sunt ambele aplicate entităţii } e_i \text{ şi } r_h < r_l \\ \text{AA}, & \text{dacă } r_h \text{ şi } r_l \text{ sunt ambele aplicate entităţii } e_i \text{ şi } r_h > r_l \\ \text{N}, & \text{dacă } r_h \text{ şi } r_l \text{ nu se pot aplica entităţii } e_i \end{cases}, \tag{3}$$

unde $ra(r_h, e_i) = T$, $ra(r_l, e_i) = T$, $1 \leq h, l \leq t,\ 1 \leq i \leq m$.

Fie $DS = (SR^t, SE^m)$ *domeniul de decizie* pentru GMORSP şi $\vec{x} = (r_1,\ r_2,\ \ldots, r_t,\ e_1,\ e_2,\ \ldots, e_m),\ \vec{x} \in DS$, o variabilă din acest domeniu. Problema GMORSP este definită astfel:

- $f_1,\ f_2, \ldots,\ f_M$ – M funcţii obiectiv, unde $f_i : DS \to \mathcal{R}$, $i = \overline{1, M}$, şi $F(\vec{x}) = \{f_1(\vec{x}), \ldots, f_M(\vec{x})\},\ \vec{x} \in DS$;
- $g_1,\ \ldots, g_J$ – J constrângeri de inegalitate, unde $g_j(\vec{x}) \geq 0$, $j = \overline{1, J}$;
- $h_1,\ \ldots, h_K$ – K constrângeri de egalitate, unde $g_k(\vec{x}) = 0$, $k = \overline{1, K}$.

Problema GMORSP este problema identificării vectorului de decizie $\vec{x} = (x_1, \ldots, x_{m+t})$ astfel încât $optimize\{F(\vec{x})\} = optimize\{f_1(\vec{x}), \ldots, f_M(\vec{x})\}$, unde $f_i : DS \to \mathcal{R}$, $i = \overline{1, M}$, $g_j(\vec{x}) \geq 0, j = \overline{1, J}$, $h_k(\vec{x}) = 0, k = \overline{1, K}$, $\vec{x} \in DS$.

Optimizarea multicriterială presupune de multe ori optimizarea unor crtierii sau obiective conflictuale. Pentru formularea GMORSP există posibilitatea de a combina diferite tipuri de obiective, adică unele care să fie maximizate, iar altele care să fie minimizate.

5.4 Definiţii ale unor probleme multicriteriale spcifice de selectare a refactorizărilor

5.4.1 Problema Multicriterială de Selectare a Unei Mulţimi de Refactorizări

Problema Multicriterială de Selectare a Unei Mulţimi de Refactorizări (*Multi-Objective Refactoring Set Selection Problem, MORSSP*) este un caz special al problemei de selectare a refactorizărilor. Definiţia ei [CC09c] urmează GMORSP (vezi Secţiunea 5.3). Se definesc două obiective conflictuale: the *costul refactorizării* şi *impactul refactorizării* asupra entităţii soft. Pentru a defini MORSSP este necesară definirea unor noţiuni şi caracteristici suplimentare.

Notaţii Particulare

Ponderea asociată fiecărei entităţi soft $e_i, 1 \leq i \leq m$, este reţinută în mulţimea $Weight = \{w_1, \ldots, w_m\}$, unde $w_i \in [0, 1]$ şi $\sum_{i=1}^{m} w_i = 1$.

Efortul pe care îl presupune procesul de transformare este convertit în cost, fiind descris de $rc : SR \times SE \to Z$,

$$rc(r_l, e_i) = \begin{cases} > 0, & \text{dacă } ra(r_l, e_i) = T \\ = 0, & \text{altfel} \end{cases},$$

unde funcţia ra este definită de formula 2 (vezi Secţiunea 5.3), $1 \leq l \leq t$, $1 \leq i \leq m$.

Modificările asupra entităţilor soft $e_i, i = \overline{1, m}$, determinate de aplicarea refactorizării $r_l, 1 \leq l \leq t$, sunt definite de funcţia $effect : SR \times SE \to Z$,

$$effect(r_l, e_i) = \begin{cases} > 0, & \text{dacă } ra(r_l, e_i) = T \text{ şi are efectul căutat asupra entităţii } e_i \\ < 0, & \text{dacă } ra(r_l, e_i) = T \text{ şi } nu \text{ are efectul căutat asupra entităţii } e_i \\ = 0, & \text{altfel} \end{cases},$$

unde funcţia ra este definită de formula 2 (vezi Secţiunea 5.3), $1 \leq l \leq t$, $1 \leq i \leq m$.

Efectul final determinat de aplicarea refactorizării $r_l, 1 \leq l \leq t$, asupra fiecărei entităţi soft $e_i, i = \overline{1,m}$, este definit de $res : SR \to Z$, $res(r_l) = \sum_{i=1}^{m} w_i \cdot effect(r_l, e_i)$, unde $1 \leq l \leq t$ şi w_i este ponderea entităţii soft e_i din SE.

Fiecare refactorizare $r_l, l = \overline{1,t}$, poate fi aplicată asupra unei submulţimi de entităţi soft, definită ca $re : SR \to \mathcal{P}(SE)$, $re(r_l) = \{\ e_{l_1}, \ldots, e_{l_q}\ \mid$ dacă $ra(r_l, e_{l_u}) = T,\ 1 \leq u \leq q,\ 1 \leq q \leq m\ \}$, unde funcţia ra este definită de formula 2 (vezi Secţiunea 5.3), $re(r_l) = SE_{r_l}, SE_{r_l} \in \mathcal{P}(SE) - \varnothing$, $1 \leq l \leq t$.

Date de Ieşire

Problema MORSSP constă în identificarea unei submulţimi de entităţi soft notată prin $ESet_l$, $ESet_l \subseteq SE_{r_l} \subseteq SE$, pentru fiecare refactorizare $r_l \in SR, l = \overline{1,t}$ astfel încât:

- următoarele obiective să fie optimizate:
  - costul refactorizării trebuie minimizat;
  - impactul refactorizării asupra entităţilor soft trebuie maximizat;
- constrângerile reprezentate de dependenţele dintre refactorizări, definite de funcţia 3 sunt satisfăcute.

În formularea MORSSP sunt definite două obiective, unul fiind reprezentat de minimizarea costului refactorizării, iar celălalt de maximizarea efectului asupra entităţilor soft. Astfel, funcţia multicriterială $F(\vec{r}) = \{f_1(\vec{r}),\ f_2(\vec{r})\}$, unde $\vec{r} = (r_1, \ldots, r_t)$ trebuie optimizată, aşa cum este descris mai jos.

Funcţia de minimizat este costul total al refactorizărilor aplicate: $minimize\{f_1(\vec{r})\} = minimize\left\{\sum_{l=1}^{t}\sum_{i=1}^{m} rc(r_l, e_i)\right\}$, unde $\vec{r} = (r_1, \ldots, r_t)$.

Al doilea obiectiv de maximizat este efectul final al refactorizărilor asupra entităţilor soft, luând în considerare ponderea entităţilor soft:

$$maximize\left\{f_2(\vec{r})\right\} = maximize\left\{\sum_{l=1}^{t} res(r_l)\right\} = maximize\left\{\sum_{l=1}^{t}\sum_{i=1}^{m} w_i \cdot effect(r_l, e_i)\right\}, \quad (4)$$

unde $\vec{r} = (r_1, \ldots, r_t)$.

Pentru a converti primul obiectiv într-o funcţie de maximizare în MORSSP, costul total este scăzut din MAX, cel mai mare cost total posibil, aşa cum este descris mai jos:

$$maximize\left\{f_1^{'}(\vec{r})\right\} = maximize\left\{MAX - \sum_{l=1}^{t}\sum_{i=1}^{m} rc(r_l, e_i)\right\}, \quad (5)$$

unde $\vec{r} = (r_1, \ldots, r_t)$. Funcţia obiectiv finală pentru MORSSP este definită prin:

$$\begin{aligned} &maximize\left\{F(\vec{r})\right\} = maximize\left\{f_1^{'}(\vec{r}),\ f_2(\vec{r})\right\} = \\ &= maximize\left\{MAX - \textstyle\sum_{l=1}^{t}\sum_{i=1}^{m} rc(r_l, e_i),\ \sum_{l=1}^{t}\sum_{i=1}^{m} w_i \cdot effect(r_l, e_i)\right\}, \end{aligned} \quad (6)$$

unde $\vec{r} = (r_1, \ldots, r_t)$.

5.4.2 Problema Multicriterială de Selectare a Unei Refactorizări

Problema Multicriterială de Selectare a Unei Refactorizări (*Multi-Objective Refactoring Single Selection Problem, MORSgSP*) reprezintă un caz particular al MORSSP (vezi Subsecţiunea 5.4.1) cu cerinţe restrânse [CCV09a]. Astfel, aspectele specifice problemei sunt izolate în *datele de ieşire*.

Date de Ieşire

MORSgSP este o problema multicriterială de identificare a refactorizării $r_l \in SR, l = \overline{1,t}$, pentru fiecare entitate soft $e_i \in SE, i = \overline{1,m}$, astfel încât:

- următoarele obiective sa fie optimizate:
  - costul refactorizării trebuie minimizat;
  - impactul refactorizărilor asupra entităţilor soft trebuie maximizat.
- constrângerile reprezentate de dependenţele dintre refactorizări, definite de funcţia 3 trebuie satisfăcute.

5.4.3 Problema Multicriterială de Selectare a Unei Secvenţe de Refactorizări

Definiţia *Problemei Multicriterială de Selectare a Unei Secvenţe de Refactorizări* (*Multi-Objective Refactoring Sequence Selection Problem, MORSqSP*) se bazează pe GMORSP (vezi Secţiunea 5.3). Scopul este identificarea unei secvenţe de refactorizări care afectează produsul soft, ţinând cont de dependenţele existente între entităţile soft şi refactorizările aplicate acestora. S-au folosit două obiective conflictuale (*costul refactorizării* şi *impactul refactorizării*) definite pentru MORSSP.

Elementele specifice problemei sunt definite ca *noţiuni particulare* necesare, iar cerinţele problemei sunt descrise ca *date de ieşire*.

Noţiuni Particulare

Definiţia 5.4.1([CC10f]) *O* ***pereche refactorizare-entitate****, notată prin* $\widehat{r_l\ e_i} = (r_l, e_i)$*, constă în refactorizarea* $r_l, 1 \leq l \leq t$*, aplicată entităţii soft* $e_i, 1 \leq i \leq m$*, astfel* $ra(r_l, e_i) = T$*.*

Definiţia 5.4.2([CC10f]) *O* ***secvenţă de refactorizări*** *este o succesiune de perechi refactorizare-entitate* $rs = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2},\ \ldots, \widehat{r_s\ e_s})$*, unde* $r_u \in SR,\ e_u \in SE,\ 1 \leq u \leq s$*. Următoarele condiţii sunt îndeplinite:*

1. $ed(e_u, e_{u+1}) \in SED, \forall\ \widehat{r_u\ e_u}, \widehat{r_{u+1}\ e_{u+1}} \in rs$*,unde* $r_u,\ r_{u+1} \in SR,\ e_u, e_{u+1} \in SE, 1 \leq u \leq s-1$*;*
2. $rd(r_u, r_{u+1}, e_u), rd(r_u, r_{u+1},\ e_{u+1}) \in SRD, \forall\ \widehat{r_u\ e_u}, \widehat{r_{u+1}\ e_{u+1}} \in rs$*,unde* $r_u, r_{u+1} \in SR,\ e_u, e_{u+1} \in SE, 1 \leq u \leq s-1$*;*
3. $rd(r_l, r_k, e_i) \in SRD, \forall\ \widehat{r_l\ e_i}, \widehat{r_k\ e_i} \in rs$*,unde* $r_l,\ r_k \in SR, e_i \in SE, 1 \leq l < k \leq t, 1 \leq i \leq m$*;*
4. $ed(e_i, e_k) \in SED, \forall\ \widehat{r_l\ e_i}, \widehat{r_l\ e_k} \in rs$*,unde* $r_l \in SR, e_i, e_k \in SE, 1 \leq i < k \leq m, 1 \leq l \leq t$*.*

O *secvenţă de refactorizări aplicată unei entităţi soft* $e_i,\ 1 \leq i \leq m$, notată prin rs_{e_i}, este descrisă prin $rs_{e_i} = (r_1,\ r_2, \ldots, r_s),\ e_i \in SE,\ 1 \leq i \leq m,\ r_u \in SR,\ 1 \leq u \leq s,\ s \in \mathcal{N}^*$. Funcţiile *ra*, *ed*, şi *rd* sunt descrise formal în Secţiunea 5.3.

Mulţimea tuturor secvenţelor de refactorizări aplicabile entităţilor soft $e_i,\ i = \overline{1,m}$*, utilizând refactorizările* $r_l,\ l = \overline{1,t}$, este definită ca:

$$SSR = \{\ rs\ \ |\ rs = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots, \widehat{r_s\ e_s}),\ r_u \in SR,\ e_u \in SE,\ 1 \leq u \leq s,\ s \in \mathcal{N}^*\ \}. \quad (7)$$

Mulţimea tuturor secvenţelor de refactorizări aplicabile unei entităţi soft $e,\ e \in SE$, este definită ca:

$$SSR_e = \{\ rs_e\ \ |\ rs_e = (r_1,\ r_2, \ldots, r_s),\ r_u \text{ este o refactorizare aplicată entităţii } e,\ 1 \leq u \leq s,\ s \in \mathcal{N},\ \}, \quad (8)$$

unde $SSR_e \in \mathcal{P}(SSR),\ e \in SE$.

Mulţimea refactorizărilor distincte care formează o secvenţă de refactorizări $rs = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots,\ \widehat{r_s\ e_s}),\ s \in \mathcal{N}$, este notată prin $SR_{rs} = \{r|\ \exists\ \widehat{r\ e}\ in\ rs,\ e \in SE,\ r \in SR\}$, $SR_{rs} \in \mathcal{P}(SR)$.

Mulţimea perechilor distincte refactorizare-entitate care formează o secvenţă de refactorizări $rs = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots, \widehat{r_s\ e_s}),\ s \in \mathcal{N}$, este notată prin $SEP_{rs} = \{\widehat{r\ e}|\ \exists\ \widehat{r\ e}\ in\ rs,\ r \in SR,\ e \in SE\}$, $SEP_{rs} \in \mathcal{P}(SSR)$.

Date de Ieşire

Obiectivele MORSqSP sunt:

1. identificarea secvenţei de refactorizări $rs = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots, \widehat{r_s\ e_s})$ aplicată produsului soft SS, unde $e_u \in SE,\ r_u \in SR,\ 1 \leq u \leq s,\ s \in \mathcal{N}$;
2. identificarea secvenţei de refactorizări $rs_e = (r_1,\ r_2, \ldots, r_s)$ aplicată unei entităţi soft e, unde $e \in SE,\ r_u \in SR,\ 1 \leq u \leq s,\ s \in \mathcal{N}$.

Multicriterialitatea MORSqSP constă în:

- obiectivele de optimizat:
  - costul refactorzării trebuie minimizat;
  - impactul refactorizării asupra entităţilor soft trebuie maximizat;
- constrângerile reprezentate de dependenţele dintre entităţile soft, definite de funcţia 1, sunt satisfăcute;
- constrângerile reprezentate de dependenţele dintre refactorizări, definite de funcţia 3, sunt satisfăcute.

Funcţia obiectiv pentru MORSqSP este similară celei din formula 6. Vectorul de decizie $\vec{r}$ este descris conform obiectivelor şi are un domeniu de decizie DS corespunzător. Astfel, pentru obiectivele propuse ca date de ieşire, avem:

- identificarea *secvenţei de refactorizări* rs *aplicată produsului soft* SS:
  - $DS = SSR$;
  - $\vec{r} = (\widehat{r_1, e_1},\ \widehat{r_2, e_2}, \ldots, \widehat{r_s, e_s})$, unde $e_u \in SE,\ r_u \in SR,\ 1 \leq u \leq s,\ s \in \mathcal{N},\ s \leq n$;
- identificarea *secvenţei de refactorizări* rs_{e_i} *aplicată entităţii soft* e_i, unde $e_i \in SE,\ 1 \leq i \leq m$:
  - $DS = SR^n,\ n \in \mathcal{N}$;
  - $\vec{r} = (r_1,\ r_2, \ldots, r_s)$, unde $e_i \in SE,\ r_u \in SR, 1 \leq i \leq m, 1 \leq u \leq s, s \leq n, s \in \mathcal{N}$.

5.4.4 Problema Multicriterială de Construire a Unui Plan de Refactorizare

Această subsecţiune descrie *Problema Multicriterială de Construire a Unui Plan de Refactorizare* (*Multi-Objective Refactoring Plan Building Problem, MORPBP*), pornind de la GMORSP (vezi Secţiunea 5.3) şi MORSqSP (vezi Subsecţiunea 5.4.3). Motivaţia şi scenariul de lucru pentru problema propusă sunt descrise în această subsecţiune. Datele de intrare şi notaţiile particulare sunt necesare pentru definirea completă a MORPBP.

Date de Intrare

Fie $SE = \{e_1, \ldots, e_m\}$ o mulţime de entităţi soft aşa cum au fost definite în Secţiunea 5.3. Mulţimea de tipuri de dependenţe existente între entităţile soft SED şi funcţia de dependenţă ed sunt similare celor descrise de formula 1 (vezi Secţiunea 5.3).

O mulţime de refactorizări alese care se pot aplica entităţilor soft din SE este $SR = \{r_1, \ldots, r_t\}$. Funcţia ra stabileşte aplicabilitatea refactorizărilor din SR asupra entităţilor soft din SE, definite prin formula 2 (vezi Secţiunea 5.3).

Mulţimea de tipuri de dependenţe existente între refactorizări SRD şi funcţia de dependenţă rd, ce evidenţiază dependenţele dintre refactorizări atunci când acestea sunt aplicate aceleaşi entităţi soft sunt descrise de formula 3 (vezi Secţiunea 5.3).

Definiţiile 5.4.1 şi 5.4.2 (vezi Subsecţiunea 5.4.3) introduc noţiunile de *pereche refactorizare-entitate* şi *secvenţă de refactorizări*. Mulţimea de secvenţe de refactorizări care se poate aplica entităţilor soft $e_i,\ i = \overline{1, m}$, folosind refactorizările $r_l,\ l = \overline{1, t}$, notată prin $SSR = \{rs_1,\ \ldots, rs_p\}$, unde $rs_k = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots, \widehat{r_{s_{rs_k}}\ e_{s_{rs_k}}}),\ k = \overline{1, p},\ e_u \in SE,\ r_u \in SR,\ 1 \leq u \leq s_{rs_k},\ p, s_{rs_k} \in \mathcal{N}$, este definită similar de formula 7 (vezi Subsecţiunea 5.4.3).

Fiecare secvenţă de refactorizări $rs_k,\ 1 \leq k \leq p$, are o *pondere de construcţie* în cadrul mulţimii de secvenţe de refactorizări SSR, care indică prioritatea secvenţelor de refactorizare

când sunt combinate între ele, fiind definite de mulţimea $SSRWeight = \{rsw_1, \ldots, rsw_p\}$, unde $rsw_k \in [0,1]$ şi $\sum_{k=1}^{p} rsw_k = 1$.

Fiecare pereche refactorizare-entitate $\widehat{r_u\ e_u},\ u = \overline{1, s_{rs_k}}$, dintre secvenţele de refactorizări participante $rs_k = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots, r_{s_{rs_k}}\widehat{e_{s_{rs_k}}})$, unde $rs_k \in SSR,\ k = \overline{1,p},\ s_{rs_k} \in \mathcal{N}$, în procesul de construire al planului de refactorizare, are asociată o *stare de integrare*. Aceasta se ataşează fiecărei perechi refactorizare-entitate la momentul construcţiei secvenţei de refactorizări.

Există două stări de integrare posibile pentru o pereche refactorizare-entitate la momentul compunerii unui plan de refactorizare descrise prin mulţimea $RStatus = \{\mathbf{M}andatory,\ \mathbf{O}ptional\}$. Funcţia care descrie starea perechilor refactorizare-entitate este definită ca $rstatus_{rs} : SEP_{rs} \rightarrow RStatus$,

$$rstatus_{rs}(\widehat{r\ e}) = \begin{cases} \text{M}, & \text{dacă } \widehat{r\ e} \text{ este } \textit{obligatorie} \text{ în construirea planului de refactorizare} \\ \text{O}, & \text{altfel} \end{cases},$$

unde $\widehat{r\ e} \in SEP_{rs}$.

Notaţii Particulare

Construirea unui plan de refactorizare bazată pe secvenţe de refactorizări constă în combinarea mai multor secvenţe de refactorizări într-o singură secvenţă de refactorizări. Secvenţa de refactorizări rezultată conţine puncte de conectare (joncţiune) între fiecare două perechi refactorizare-entitate, care fac parte din aceeaşi secvenţă de refactorizări sau nu.

Definiţia 5.4.3([CC10f]) *Fie* $rs = (\widehat{r_1\ e_1},\ \widehat{r_2\ e_2}, \ldots, \widehat{r_s\ e_s}), rs \in SSR$, *o secvenţă de refactorizări,* $\widehat{r_u\ e_u}$,$r_{u+1}\widehat{e_{u+1}}$ *două perechi refactorizare-entitate consecutive în* rs, *unde* $1 \leq u \leq s-1$ *şi* $\widehat{r\ e}$ *este o pereche refactorizare-entitate, unde* $e \in SE, r \in SR, ra(r,e) = T$ *şi* $\widehat{r\ e}$ *nu este în* rs. *Atunci:*

1. *perechea refactorizare-entitate* $\widehat{r_u\ e_u},\ 1 \leq u \leq s$, *are un* ***punct de joncţiune înainte****, notat prin* ${}_b(\widehat{r_u\ e_u})$, *dacă există o pereche refactorizare-entitate* $\widehat{r\ e}$ *care poate fi inserată înaintea perechii* $\widehat{r_u\ e_u}$ *în secvenţa de refactorizări* rs, *unde* rs *îmbunătăţită cu* $\widehat{r\ e}$ *este* $rs' = (\ldots, \widehat{r\ e},\ \widehat{r_u\ e_u}, \ldots)$ *şi următoarele condiţii sunt îndeplinite:*
   (a) $ed(e, e_u) \in SED$, *unde* $e,\ e_u \in SE,\ 1 \leq u \leq s$;
   (b) $rd(r, r_u) \in \{B,\ AB\} \subseteq SRD$, *unde* $r,\ r_u \in SR,\ 1 \leq u \leq s$.
2. *cele două perechi refactorizare-entitate* $\widehat{r_u\ e_u}$ *şi* $r_{u+1}\widehat{e_{u+1}},\ 1 \leq u \leq s-1$, *au un* ***punct de joncţiune de mijloc****, notat prin* $(\widehat{r_u\ e_u})_m(r_{u+1}\widehat{e_{u+1}})$, *dacă există o pereche refactorizare-entitate* $\widehat{r\ e}$ *care se poate insera între perechile* $\widehat{r_u\ e_u}$ *şi* $r_{u+1}\widehat{e_{u+1}}$ *în secvenţa de refactorizări* rs, *unde* rs *îmbunătăţită cu* $\widehat{r\ e}$ *este* $rs' = (\ldots, \widehat{r_u\ e_u}, \widehat{r\ e}, r_{u+1}\widehat{e_{u+1}}, \ldots)$ *şi următoarele condiţii sunt îndeplinite:*
   (a) $ed(e_u, e),\ ed(e, e_{u+1}) \in SED$, *unde* $e_u,\ e,\ e_{u+1} \in SE,\ 1 \leq u \leq s-1$;
   (b) $rd(r_u, r),\ rd(r, r_{u+1}) \in \{B,\ AB\} \subseteq SRD$, *unde* $r_u,\ r,\ r_{u+1} \in SR,\ u = \overline{1, s-1}$.
3. *o pereche refactorizare-entitate* $\widehat{r_u\ e_u},\ 1 \leq u \leq s$, *are un* ***punct de joncţiune după****, notat prin* $(\widehat{r_u\ e_u})_a$, *dacă există o pereche refactorizare-entitate* $\widehat{r\ e}$ *care poate fi adăugată după perechea* $\widehat{r_u\ e_u}$ *în secvenţa de refactorizări* rs, *unde* rs *îmbunătăţită cu* $\widehat{r\ e}$ *este* $rs' = (\ldots,\ \widehat{r_u\ e_u},\ \widehat{r\ e},\ \ldots)$ *şi următoarele condiţii sunt îndeplinite:*
   (a) $ed(e_u, e) \in SED$, *unde* $e_u,\ e \in SE,\ 1 \leq u \leq s$;
   (b) $rd(r_u, r) \in \{B,\ AB\} \subseteq SRD$, *unde* $r_u,\ r \in SR,\ 1 \leq u \leq s$.

O *mulţime de puncte de joncţiune refactorizare-entitate pentru o secvenţă de refactorizări* rs este definită ca:
$REP_{rs}^{jp} = \{{}_b(\widehat{r_u\ e_u}) |\ \widehat{r\ e}$ *este inserata inainte* $\widehat{r_u\ e_u},\ 1 \leq u \leq s\} \cup \{(\widehat{r_u\ e_u})_m(r_{u+1}\widehat{e_{u+1}}) |\ \widehat{r\ e}$ *este*

inserata intre $\widehat{r_u\ e_u}$ *si* $r_{u+1}\widehat{\ e_{u+1}}$, $1 \le u \le s-1\} \cup \{(\widehat{r_u\ e_u})_a |\ \widehat{r\ e}$ *este inserata dupa* $\widehat{r_u\ e_u}$, $1 \le u \le s\}$, fiind construită la momentul construirii secvenţelor de refactorizări.

Definiţia 5.4.4([CC10f]) *Fie* $SSR_{rp} = \{rs_1, \ \ldots, rs_q\}$ *o mulţime de secvenţe de refactorizări,* $REP^{jp}_{rs_1}$, $REP^{jp}_{rs_2}, \ldots,$ $REP^{jp}_{rs_q}$ *mulţimile de puncte de joncţiune corespunzătoare secvenţelor de refactorizări* rs_k, *unde* $1 \le k \le q$, *unde* $SSR_{rp} \in \mathcal{P}(SSR)$, $rs_k = (\widehat{r_1\ e_1}, \widehat{r_2\ e_2}, \ldots, r_{s_{rs_k}}\widehat{\ e_{s_{rs_k}}})$, $k = \overline{1,q}$, $e_u \in SE$, $r_u \in SR$, $1 \le u \le s_{rs_k}$, $s_{rs_k}, q \in \mathcal{N}$.

Un **plan de refactorizare** este o secvenţă de refactorizări $rp = (\widehat{r_1\ e_1}, \widehat{r_2\ e_2}, \ldots, r_{s_{rp}}\widehat{\ e_{s_{rp}}})$ construită prin navigarea între perechile refactorizare-entitate $\widehat{r_u\ e_u}$, $1 \le u \le s_{rs_k}$, $k = \overline{1,q}$, $q \in \mathcal{N}$ ale secvenţelor de refactorizări participante rs_k din mulţimea SSR_{rp}. Atunci, pentru fiecare pereche refactorizare-entitate $\widehat{r_u\ e_u} \in rp$, unde $1 \le u \le s_{rp}$, există un punct de joncţiune jp, unde $jp \in REP^{jp}_{rs_k}$, astfel încât jp este punctul de joncţiune care a introdus perechea refactorizare-entitate $\widehat{r_u\ e_u}$ în planul de refactorizare rp, unde $1 \le k \le q$.

Date de Ieşire

MORPBP este problema identificării planului de refactorizare $rp = (\widehat{r_1\ e_1}, \widehat{r_2\ e_2}, \ldots, r_{s_{rp}}\widehat{\ e_{s_{rp}}})$, $s_{rp} \in \mathcal{N}$, folosind mulţimea de secvenţe de refactorizări SSR_{rp}, astfel încât:

- următoarele obiective sunt optimizate:
  - costul refactorizării trebuie minimizat;
  - impactul refactorizării asupra entităţilor soft trebuie maximizat;
- constrângerile reprezentate de dependenţele dintre entităţile soft, definite de funcţia 1, sunt satisfăcute;
- constrângerile reprezentate de dependenţele dintre refactorizări, definite de funcţia 3, sunt satisfăcute.

În MORPBP se optimizează două obiective: *costul refactorizăii* şi *impactul refactorizării* asupra entităţilor soft afectate. Funcţia multicriterială

$$optimize\{F(\vec{rp})\} = optimize\{f_1(\vec{rp}),\ f_2(\vec{rp})\}, \tag{9}$$

unde $\vec{rp}$= $(\widehat{r_1\ e_1}, \widehat{r_2\ e_2}, \ldots, r_{s_{rp}}\widehat{\ e_{s_{rp}}})$, $s_{rp} \in \mathcal{N}$, este optimizată, fiind definită similar în Secţiunea 5.3.

Formularea curentă pentru MORPBP prezintă costul refactorizării ca şi obiectiv şi nu ca şi constrângere, aşa cum sunt definite dependenţele dintre entităţile soft de formula 1 (vezi Secţiunea 5.3) şi dependenţele dintre refactorizări descrise de funcţia 3.

Domeniul de decizie este $DS = SSR$, iar vectorul de decizie este $\vec{rp}$= $(\widehat{r_1\ e_1}, \widehat{r_2\ e_2}, \ldots, r_{s_{rp}}\widehat{\ e_{s_{rp}}})$, $s_{rp} \in \mathcal{N}$, conţine secvenţele de refactorizări obţinute prin navigarea mulţimii de secvenţe de refactorizări SSR_{rp}, $SSR_{rp} \in \mathcal{P}(SSR)$ propusă.

5.5 Concluzii şi direcţii de cercetare viitoare

Selectarea refactorizărilor adecvate pentru produse soft de diferite dimensiuni este problema de cercetare investigată în acest capitol. Dependenţele dintre entităţile soft şi dependenţele dintre refactorizări reprezintă elementele de bază care dirijează cercetarea.

Câteva dintre rezultatele originale prezentate în acest capitol au fost publicate în articolele [CC09b, CC09a, CC10a, CC10d]. Capitolul se bazează pe rezultatele originale publicate în [CCV09a, CCV09b, CC09c, CC09e, CC10f].

Câteva direcţii de cercetare viitoare: definirea problemei de construire a planului de refactorizare folosind abordarea aplicării refactorizărilor în paralel, cu notaţii noi pentru punctele multi-joncţiune care furnizează puncte de multi-conexiune între diferite secvenţe de refactorizări; formularea unor probleme multicriteriale care folosesc costul refactorizării ca şi constrângere şi nu ca obiectiv; formularea unor probleme multicriteriale care folosesc mai multe obiective de optimizat.

6 Abordare evolutivă a problemei selectării refactorizărilor

Acest capitol investighează *Problema de Selectare Multicriterială a Unei Mulţimi de Refactorizări* şi cazul particular al acesteia, *Problema de Selectare Multicriterială a Unei Refactorizări*. S-au studiat diferiţi algoritmi genetici şi reprezentări ale soluţiilor. *Problema de Selectare Multicriterială a Unei Mulţimi de Refactorizări* este modelată printr-o singură funcţie obiectiv, cu două obiective conflictuale de optimizat. De asemenea, sunt analizate rezultatele abordărilor propuse.

6.1 Reprezentări ale soluţiei evolutive pentru Problema Multicriterială de Selectare a Unei Mulţimi de Refactorizări

Abordarea MORSSP prezentată aici explorează dependenţele dintre refactorizări. Cele două obiective de optimizat sunt minimizarea costului şi maximizarea impactului refactorizării, iar constrângerile de satisfăcut sunt dependenţele dintre refactorizări.

Pentru rezolvarea problemei MORSSP s-a folosit *metoda sumei ponderate* [Kd05]. Funcţia obiectiv de maximizat $F(\vec{r})$ este modelată după principiul sumei ponderate cu două obiective de optimizat. Astfel, $maximize\left\{F(\vec{r})\right\} = maximize\left\{f_1(\vec{r}), f_2(\vec{r})\right\}$, este rescrisă matematic prin $maximize\left\{F(\vec{r})\right\} = \alpha \cdot f_1(r) + (1-\alpha) \cdot f_2(r)$, unde $0 \leq \alpha \leq 1$ şi $\vec{r}$ este variabila de decizie.

A fost propus modelul evolutiv steady–state împreună cu un algoritm genetic adaptat şi funcţia obiectiv ponderată, prezentată în [CC09b, CC10a].

Algoritmul genetic studiat foloseşte o reprezentare a soluţiei *bazată pe rafactorizări* pentru selectarea unei mulţimi de refactorizări, algoritm notat prin *RSSGARef*, iar reprezentarea *bazată pe entităţi* este folosită de algoritmul notat prin *RSSGAEnt*.

6.2 Reprezentări studiate ale soluţiei evolutive

6.2.1 Reprezentarea soluţiei bazată pe refactorizări

Pentru algoritmul *RSSGARef*, reprezentarea soluţiei este prezentată în [CC09c]. Vectorul de decizie $\vec{S} = (S_1, \ldots, S_t)$, unde $S_l \in \mathcal{P}(SE), 1 \leq l \leq t$, descrie entităţile soft care pot fi transformate folosind mulţimea de refactorizări SR. Elementul S_l de pe poziţia l a vectorului soluţie reprezintă o mulţime de entităţi soft care pot fi transformate prin aplicarea refactorizării l din SR, unde orice $e_{l_u} \in SE_{r_l}, e_{l_u} \in S_l \in \mathcal{P}(SE), 1 \leq u \leq q, 1 \leq q \leq m, 1 \leq l \leq t$. Aceasta înseamnă că este posibilă aplicarea de mai multe ori a aceleaşi refactorizări unor entităţi soft diferite. Operatorii genetici folosiţi de această abordare sunt încrucişarea şi mutaţia.

6.2.2 Reprezentarea soluţiei bazată pe entităţi

Algoritmul *RSSGAEnt* foloseşte reprezentarea soluţiei prezentată în [CC09e], unde vectorul de decizie $\vec{S} = (S_1, \ldots, S_m), S_i \in \mathcal{P}(SR), 1 \leq i \leq m$ descrie refactorizările care pot fi aplicate pentru transformarea mulţimi de entităţi soft SE. Elementul S_i de pe poziţia i a vectorului soluţie reprezintă o mulţime de refactorizări care se pot aplica entităţii soft i din SE, unde fiecare $e_{l_u} \in SE_{r_l}, S_{r_l} \in \mathcal{P}(SR), 1 \leq u \leq q, 1 \leq q \leq m, 1 \leq l \leq t$. Aceasta înseamnă că o refactorizare se poate aplica de mai multe ori unor entităţi soft diferite. Operatorii genetici folosiţi de această abordare sunt încrucişarea şi mutaţia.

6.3 Studiu de caz: Problema simulării funcţionării unei reţele locale

Algoritml genetic adaptat, prezentat în [CC09b, CC10a], este aplicat unei versiuni simplifcate a codului sursă al problemei de simulare a funcţionării unei reţele de calculatoare (vezi Subsecţiunea 1.11.1). Informaţiile relevante referitoare la codul sursă sunt extrase în mulţimea de entităţi soft: $SE = \{c_1, \ldots, c_5, a_1, \ldots, a_5, m_1, \ldots, m_{13}\}$, $|SE| = 23$. Transformările alese sunt refactorizări care sunt aplicate claselor, atributelor şi metodelor: *RenameMethod, ExtractSuperClass, PullUpMethod, MoveMethod, EncapsulateField, AddParameter*. Acestea formează mulţimea de refactorizări $SR = \{r_1, \ldots, r_6\}$ în continuare. Ponderile entităţilor soft sunt grupate în mulţimea $Weight$, prezentate în Tabela 3, unde $\sum_{i=1}^{23} w_i = 1$.

Relaţia de dependenţă dintre refactorizări, descrisă prin funcţia rd şi impactul final al fiecărei refactorizări descris de funcţia res sunt prezentate în Tabela 3. Calculul valorilor funcţiei res pentru fiecare refactorizare este bazată pe ponderea fiecărei entităţi soft afectate, aşa cum este descris în Secţiunea 5.3. Aplicabilitatea refactorizărilor (funcţia ra) pentru fiecare entitate soft din SE este exprimată printr-o valoare nenulă a costului refactorizării, prezentat de Tabela 3. Astfel, pentru entitatea soft c_3 şi refactorizarea r_2, notaţiile $\surd/2$ înseamnă că refactorizarea se poate aplica entităţii precizate ($\surd$, i.e., $ra(r_2, c_3) = T$), având costul transformării 2.

Fiecare entitate soft poate fi transformată de refactorizări specifice, altfel valoarea funcţiei de cost este 0. De exemplu, refactorizările r_1, r_3, r_4, r_6 pot fi aplicate metodelor m_1, m_4, m_7, m_{10}, m_{13}. Pentru metodele speciale (constructorii), refactorizările ca $pullUpMethod$ (r_3) şi $moveMethod$ (r_4) nu se pot aplica. Valoarea funcţiei cost rc este calculată prin numărul de transformări necesare pentru a aplica refactorizarea. Astfel, diferite refactorizări au cost de aplicare diferit atunci cand sunt folosite în cazul unor entităţi soft înrudite.

6.3.1 Strategia de refactorizare propusă

O strategie de refactorizare pentru *Problema simulării funcţionării unei reţele* este prezentată mai jos. Pornind de la *dificultăţile* prezentate pentru ierarhia de clase corespunzătoare, au fost identificate trei categorii de transformări. Pentru fiecare dintre ele, câteva obiective care ar trebui realizate prin refactorizare sunt definite.

1. *gestionarea informaţiei* (ascunderea datelor, coeziunea datelor):
   (a) controlul accesului la atribute (refactorizarea *EncapsulateField*);
2. *gestionarea comportamentului* (definirea metodelor, coeziunea metodelor):
   (a) modificarea semnăturii metodei într-un context nou (refactorizarea *AddParameter*);
   (b) creşterea nivelului de expresivitate a identificatorului de metodelor prin modificarea numelui acestora (refactorizarea *RenameMethod*);
   (c) creşterea coeziunii metodelor din cadrul claselor (refactorizările *MoveMethod* şi *PullUpMethod*);
3. *abstractizarea la nivelul ierarhiei de clase* (generalizarea claselor, specializarea claselor):
   (a) creşterea nivelului de abstractizare la nivelul ierarhiei de clase prin generalizare (refactorizarea *ExtractSuperClass*).

6.4 Experimente rractice ale algoritmilor *RSSGARef* şi *RSSGAEnt*

Algoritmul a fost rulat de 100 ori şi s-au analizat cea mai bună, media şi cea mai slabă valoare a valorilor de fitness. Parametrii folosiţi de abordarea evolutivă sunt: probabilitate de mutaţie 0.7 şi probabilitate de încrucuşare 0.7. S-au folosit diferite valori ale numărului

(a) Dependenţele dintre refactorizări (rd) şi impactul final (res) al aplicării acestora mulţimii de entităţi soft (SE)

rd	r_1	r_2	r_3	r_4	r_5	r_6
r_1	N		B			AA
r_2		N	B			
r_3	A	A	N	N		
r_4			N	N		
r_5					N	
r_6	AB					N
res	0.4	0.49	0.63	0.56	0.8	0.2

(b) Costul refactorizării (rc) şi aplicabilitatea refactorizărilor asupra entităţilor soft. Ponderea fiecărei entităţi soft ($Weight$)

rc	r_1	r_2	r_3	r_4	r_5	r_6	$Weight$
c_1		$\surd/1$					0.1
c_2		$\surd/1$					0.08
c_3		$\surd/2$					0.08
c_4		$\surd/2$					0.07
c_5		$\surd/1$					0.07
a_1					$\surd/4$		0.04
a_2					$\surd/5$		0.03
a_3					$\surd/5$		0.03
a_4					$\surd/5$		0.05
a_5					$\surd/5$		0.05
m_1	$\surd/1$		$\surd/0$	$\surd/0$		$\surd/1$	0.04
m_2	$\surd/3$		$\surd/1$	$\surd/1$		$\surd/3$	0.025
m_3	$\surd/5$		$\surd/1$	$\surd/1$		$\surd/5$	0.025
m_4	$\surd/1$		$\surd/0$	$\surd/0$		$\surd/1$	0.04
m_5	$\surd/1$		$\surd/1$	$\surd/1$		$\surd/1$	0.025
m_6	$\surd/1$		$\surd/1$	$\surd/1$		$\surd/1$	0.025
m_7	$\surd/1$		$\surd/0$	$\surd/0$		$\surd/1$	0.04
m_8	$\surd/2$		$\surd/1$	$\surd/1$		$\surd/2$	0.025
m_9	$\surd/1$		$\surd/1$	$\surd/1$		$\surd/1$	0.025
m_{10}	$\surd/1$		$\surd/0$	$\surd/0$		$\surd/1$	0.04
m_{11}	$\surd/2$		$\surd/1$	$\surd/1$		$\surd/2$	0.025
m_{12}	$\surd/1$		$\surd/1$	$\surd/1$		$\surd/1$	0.025
m_{13}	$\surd/1$		$\surd/0$	$\surd/0$		$\surd/1$	0.04
							$\sum_{i=1}^{23} w_i = 1$

Tabela 3: Datele de intrare pentru studiul de caz: *Problema simulării funcţionării unei reţele*

de generaţii şi indivizi, ca: număr de generaţii 10, 50, 100 şi 200 şi număr de indivizi 20, 50, 100 şi 200.

6.4.1 Experimentul 1: Ponderi egale pentru cost şi impactul refactorizării ($\alpha = 0.5$)

Această subsecţiune prezintă rezultatele obţinute pentru experimentul care foloseşte ponderi egale ($\alpha = 0.5$) pentru algoritmii *RSSGARef* şi *RSSGAEnt* [CC09c, CC09b, CC09a].

Rulările pentru diferite generaţii, arată îmbunătăţirea celui mai bun cromozom. Pentru experimentle derulate, cel mai bun individ a fost obţinut pentru algoritmul *RSSGARef* după o evoluţie de 200 generaţii cu o populaţie de 20 indivizi, având valoarea funcţiei de fitness de 0.4793. Aceata înseamnă că în populaţiile cu număr redus de cromozomi, diversitatea redusă a populaţiei poate induce o competiţie puternică în comparaţie cu populaţiile cu număr mare de cromozomi, unde divesitatea populaţiei produce indivizi de calitate apropiată.

Pentru ambele reprezentări ale soluţiei, populaţiile cu număr redus de cromozomi îşi păstrează calitatea cromozomilor, prin producerea tuturor celor 100 cei mai buni indivizi cu valoarea funcţiei de fitness mai bună ca valoarea de referinţă, 0.41, pentru reprezentarea bazată pe refactorizări şi 0.15 pentru reprezentarea bazată pe entităţi. Rezultatele obţinute de reprezentarea bazată pe refactorizări sunt mai bune ca şi valoare a funcţiei de fitness. Impactul asupra codului sursă folosit ca studiu de caz după aplicarea algoritmilor *RSSGARef* şi *RSSGAEnt* sunt prezentate aici.

6.4.2 Experimentul 2: Pondere mai mare a impactului refactorizării ($\alpha = 0.3$)

Această subsecţiune prezintă rezultatele primului experiment care foloseşte ponderi diferite, impactul final (funcţia res) având o pondere mai mare decât costul refactorizărilor aplicate (funcţia rc) şi fiind prezentate în [CC09b, CC09a]. Funcţia ponderată folosită pentru algoritmii *RSSGARef* şi *RSSGAEnt* aplicaţi este $F(\vec{r}) = 0.3 \cdot f_1(\vec{r}) + 0.7 \cdot f_2(\vec{r})$, unde $\vec{r} = (r_1, \ldots, r_t)$.

Experimentul pentru algoritmul *RSSGARef Algorithm* a arătat rezultate bune pentru toate cele 100 de rulări, ca şi calitate şi număr de indivizi ai populaţiilor studiate şi număr de generaţii de evoluţie. S-a observat o tendinţă de a crea indivizi mai buni în populaţiile cu număr redus de indivizi, aşa cum a fost observat în experimentul pentru $\alpha = 0.5$.

Soluţiile pentru populaţiile cu 20 indivizi pentru numărul de generaţii studiat îşi păstrează calitatea, numărul de cromozomi eligibili fiind mai ridicat decât orice populaţie folosită în experiment. Impactul asupra codului sursă folosit ca şi studiu de caz a fost analizat după aplicarea algoritmilor *RSSGARef* şi *RSSGAEnt*.

6.4.3 Experimentul 3: Pondere mai mică a impactului refactorizării ($\alpha = 0.7$)

Această subsecţiune prezintă un al doilea experiment derulat pentru ponderi diferite ale celor două obiective, cu $\alpha = 0.7$. Aceasta înseamnă că costul refactorizării (funcţia rc) este mai important decât impactul final al refactorizării (funcţia res) asupra entităţilor soft afectate [CC09a]. Funcţia de fitness corespunzătoare algoritmilor *RSSGARef* şi *RSSGAEnt* este $F(\vec{r}) = 0.7 \cdot f_1(\vec{r}) + 0.3 \cdot f_2(\vec{r})$, unde $\vec{r} = (r_1, \ldots, r_t)$.

Cel mai bun individ obţinut pentru algoritmul *RSSGARef* a fost obţinut pentru o evouţie de 200 generaţii a unei populaţii de 20 cromozomi, cu cea mai bună valoare a funcţiei de fitness de 0.61719 (toţi indivizii sunt mai buni decât 0.53). Pentru algoritmul *RSSGAEnt* cel mai bun cromozom înregistrat a avut valoarea funcţiei de fitness de 0.16862, fiind obţinut pentru o evoluţie de 50 generaţii pentru o populaţie de 20 indivizi (cu 98 cromozomi mai buni decât 0.155).

Pentru algoritmul *RSSGARef*, rezultatele experimentului arată că sunt aproapiate de cele ale experimentului pentru $\alpha = 0.3$. Astfel, cei mai buni cromozomi pentru populaţiile cu 20 şi 200 indivizi acoperă intervalul de valori pentru funcţia de fitness de $(0.54, 0.63)$, iar pentru algoritmul *RSSGAEnt* intervalul corespunzător celui mai bun individ este mai mic $(0.15, 0.17)$, valorile funcţiei de fitness fiind distribuite în întregul interval, cu câteva excepţii.

Faţă de experimentele derulate pentru algoritmul *RSSGAEnt*, diversitatea în cadrul populaţiilor se păstrează atât pentru cel mai bun cât şi cel mai slab cromozom. Cel mai slab cromozom înregistrat pentru o evoluţie de 50 generaţii şi o populaţie de 20 indivizi a avut valoarea funcţiei de fitness de 0.12515 (27 cromozomi cu fitness mai mic decât 0.13).

De asemenea, a fost discutată aplicarea algoritmilor *RSSGARef* şi *RSSGAEnt* asupra codului sursă din studiul de caz discutat.

6.5 Discuţie asupra aplicării algoritmilor de rezolvare a Problemei Multicriteriale de Selectare a Unei Mulţimi de Refactorizări

Această secţiune prezintă rezultatele obţinute după aplicarea algoritmilor *RSSGARef* şi *RSSGAEnt* prezentaţi în Secţiunea 6.4, pentru trei valori diferite ale parametrului α, adică 0.3, 0.5 şi 0.7, cu scopul de a maximiza funcţia ponderată care optimizează costul refactorizării şi impactul acesteia asupra entităţilor soft transformate.

6.6 Analiza rezultatelor obţinute pentru Rezolvarea Problemei Multicriteriale de Selectare a Unei Mulţimi de Refactorizări

Această secţiune analizează soluţiile propuse pentru reprezentările soluţiilor bazate pe refactorizări şi entităţi ale algoritmilor aplicaţi (vezi Subsecţiunile 6.2.1 şi 6.2.2). Ambele reprezentări ale soluţiei identifică o mulţime de refactorizări pentru fiecare entitate soft asupra căreia acestea se pot aplica.

Dimensiunea cromozomului în abordarea bazată pe refactorizări este 6, adică numărul de refactorizări studiate, iar pentru abordarea bazată pe entităţi, individul are 23 gene. Refactorizările recomandate de experimentele derulate nu conturează o strategie de refactorizare omogenă pentru nici una dintre reprezentările studiate.

Cel mai bun individ obţinut pentru abordarea bazată pe refactorizări (algoritmul *RSSGARef*), a fost înregistrat pentru o evoluţie de 200 generaţii cu o populaţie de 20 cromozomi, având valoarea funcţiei de 0.4793, iar pentru abordarea bazată pe entităţi (algoritmul *RSSGAEnt*) s-a înregistrat cel mai bun cromozom pentru o evoluţie de 20 generaţii şi o populaţie de 20 indivizi, cu valoarea funcţiei de fitness de 0.17345. Aceste soluţii sunt transpuse dintr-o reprezentare în alta, ceea ce artă că structura lor poate fi comparată, iar eficienţa lor evaluată.

Ideea care rezultă din derularea acestor experimente este că populaţiile cu număr mai mic de indivizi produc indivizi mai buni (ca număr, calitate şi timp) decât cele cu număr mai mare, ceea ce poate fi determinat de diversitatea redusă în cadrul acestor populaţii. Un număr mare de gene în cadrul structurii individului determină o calitate mai slabă în reprezentarea bazată pe entităţi folosită aici.

Tabela 4 prezintă soluţiile obţinute pentru reprezentările studiate şi obiectivele atinse de fiecare dintre acestea. Numărul de obiective realizate este calculat pe baza prezenţei refactorizărilor recomandate între genele cromozomilor analizaţi.

Reprezentarea soluţiei	**Valoarea** α	**Cel mai bun crom.** (dim. pop./ nr. gen.)	**Cel mai bun fitness**	**Timp de execuţie** (secunde)	**Numărul de obiective realizate (%)**				
					Atribute *(1a)*	**Metode (2)**			**Ierarhia de clase** *(3a)*
						(2a)	*(2b)*	*(2c)*	
Bazată pe refactorizări	0.3	20c/200g	0.33587	32	0	0	0	50	100
	0.5	20c/200g	0.4793	36	60	50	50	50	50
	0.7	20c/200g	0.61719	37	0	0	50	100	50
Bazată pe entităţi	0.3	20c/200g	0.19023	61	80	50	100	100	50
	0.5	20c/200g	0.17345	75	40	0	50	100	100
	0.7	20c/50g	0.16862	19	0	50	100	100	100

Tabela 4: Cei mai buni cromozomi obţinuţi pentru reprezentările bazate pe refactorizări şi entităţi, pentru diferite valori ale parametrului α: 0.3, 0.5 şi 0.7

6.7 Abordare evolutivă a soluţiei pentru Problema Multicriterială de Selectare a Unei Refactorizări

6.7.1 Abordare bazată pe algoritmi genetici

Similar MORSSP, MORSgSP investighează dependenţele exitente între refactorizări pentru a determina cea mai potrivită refactorizare pentru fiecare entitate soft. Cele două obiective de optimizat sunt aceleaşi, adică minimizarea costului refactorizării şi maximizarea impactului acesteia. Constrângerile de satisfăcut sunt dependenţele dintre refactorizări. Abordarea curentă foloseşte metoda sumei ponderate pentru descrierea funcţiei de optimizat.

Algoritmul genetic propus foloseşte reprezentarea soluţiei bazată pe entităţi pentru problema studiată, fiind notat prin *RSgSGAEnt*. Algoritmul genetic propus în [CC10d] a fost aplicat asupra unei versiuni simplificate a studiului de caz prezentat în Subsecţiunea 1.11.1.

6.7.2 Reprezentarea soluţiei bazată pe entităţi

În reprezentarea soluţiei pentru algoritmul *RSgSGAEnt* prezentată în [CC10d], vectorul de decizie $\vec{r} = (r_1, \ldots, r_m)$, $r_i \in SR, 1 \leq i \leq m$, identifică refactorizările care se pot aplica pentru a transforma o mulţime de entităţi SE. Elementul r_i de pe poziţia i a vectorului soluţie reprezintă refactorizarea care trebuie aplicată entităţii soft i din SE, unde $e_i \in SE_{r_i}, 1 \leq i \leq m$. Operatorii genetici folosiţi de această abordare sunt încrucişarea şi mutaţia.

6.8 Experimente practice ale algoritmului *RSgSGAEnt*

Algoritmul *RSgSGAEnt* a fost aplicat de 100 ori şi cea mai bună, cea mai slabă şi media valorilor de fitness a fost înregistrată. Parametrii folosiţi de abordarea evolutivă au fost: propabilitatea de mutaţie 0.7 şi probabilitatea de încrucişare 0.7. Experimentele includ rulări ale algoritmului pentru 10, 50, 100 şi 200 generaţii cu populaţii de 20, 50, 100 şi 200 indivizi. Experimentele derulate au lucrat pe diferite valori ale parametrului α: 0.3, 0.5 şi 0.7. Astfel, funcţia de fitness este $F(\vec{r}) = \alpha \cdot f_1(\vec{r}) + (1 - \alpha) \cdot f_2(\vec{r})$, unde $\vec{r} = (r_1, \ldots, r_m)$.

6.8.1 Experimentul 1: Ponderi egale pentru cost şi impactul refactorizării ($\alpha = 0.5$)

Experimentul cu ponderi egale, pentru $\alpha = 0.5$, este studiat în această subsecţiune [CC10d].

În experimentele cu o evoluţie de 50 generaţii pentru populaţii de 200 de cromozomi, cea mai mare valoare a funcţiei de fitness a fost 0.3455 (cu 38 indivizi cu fitness > 0.33), iar pentru experimentele cu 200 de generaţii evoluţie pentru populaţii de 20 de indivizi, cel mai bun cromozom înregistrat a avut valoarea funcţiei de fitness de 0.3562 (cu 96 indivizi cu fitness > 0.33), care este cea mai mare valoare a funcţiei de fitness din experimentul curent.

Cel mai slab cromozom din toate rulările a fost obţinut pentru o populaţie de 200 cromozomi după o evoluţie de 50 generaţii, având valorea funcţiei de fitness 0.27005 (87 cromozomi cu fitness < 0.283), iar pentru populaţiile de 20 cromozomi, după o evoluţie de 200 generaţii, cel mai slab individ a avut valoarea funcţiei de fitness 0.2772 (doar 11 indivizi având fitness < 0.283). De asemenea, este investigat impactul asupra codului sursă din studiul de caz după aplicarea algoritmului *RSgSGAEnt*.

6.8.2 Experimentul 2: Pondere mai mare a impactului refactorizării ($\alpha = 0.3$)

Primul experiment care foloseşte ponderi diferite, în care impactul refactorizării (funcţia res) are o relevanţă mai mare decât costul implicat (funcţia rc) de aplicarea refactorizărilor [CC10d] este studiat în această subsecţiune.

Cel mai bun cromozom din întregul experiment a fost obţinut după o evoluţie de 100 generaţii pentru o populaţie de 200 cromozomi, având valoarea funcţiei de fitness 0.25272 (cu 69 indivizi cu fitness > 0.233), iar cel mai bun cromozom obţinut după o evoluţie de 10 generaţii pentru o populaţie de 200 indivizi, a avut valoarea funcţiei de fitness 0.24462 (34 cromozomi cu fitness > 0.233).

Cel ma slab individ obţinut în cadrul acestui experiment a avut valori asemănătoare valorilor medii. Pentru populaţiile de 200 cromozomi după o evoluţie de 10 generaţii, cel mai slab individ a avut valoara funcţiei de fitness 0.18495 (15 indivizi cu fitness mai slab decât 0.195), iar pentru o evoluţie de 200 generaţii pentru populaţiile cu 100 indivizi, cel mai slab cromozom are valoarea funcţiei de fitness 0.18907 (33 indivizi cu fitness mai slab decât 0.195).

Gruparea cromozomilor eligibili pentru populaţiile de 50, 100 şi 200 indivizi pentru număr redus de generaţii este vizibil. Soluţiile pentru populaţiile cu 20 indivizi, pentru fiecare număr de evoluţii studiat, îşi păstrează calitatea, cu număr ridicat de cromozomi eligibili. De asemenea, este prezentat impactul asupra codului sursă din studiu de caz investigat, după aplicarea algoritmului *RSgSGAEnt*.

6.8.3 Experimentul 3: Pondere mai mică a impactului refactorizării ($\alpha = 0.7$)

Al doilea experiment care foloseşte ponderi diferite, unde costul refactorizării (funcţia rc) are o relevanţă mai mare decât impactul refactorizării (funcţia res) asupra entităţilor soft afectate, este prezentat în [CC10d].

În rulările pentru o evoluţie de 200 generaţii pentru populaţii de 100 de indivizi, cea mai mare valoare a funcţiei de fitness a fost 0.44919 (75 indivizi cu fitness > 0.425), iar în evoluţiile de 50 generaţii pentru populaţii de 50 cromozomi, cel mai bun cromozom a avut valoarea funcţiei de fitness 0.45757 (65 cromozomi cu fitness > 0.425), ceea ce este cel mai bun fitness obţinut în cadrul experimentului curent. Impactul aplicării algoritmului *RSgSGAEnt* asupra codului sursă din studiu de caz este analizat.

6.8.4 Discuţie asupra aplicării algoritmului *RSgSGAEnt* pentru rezolvarea Problemei Multiriteriale de Selectare a Unei Refactorizări

Rezultatele abordării propuse în Secţiunea 6.7 pentru trei valori diferite ale parametrului α (0.3, 0.5 şi 0.7), sunt prezentate pe scurt şi discutate în această subsecţiune.

În populaţiile cu număr redus de cromozomi, diversitarea redusă induce o competenţie strânsă, în comparaţie cu populaţiile de mari dimensiuni, unde diversitatea produce indivizi de calitate apropiată ca şi valoare a funcţiei de fitness. Aşa cum au arătat-o experimentele desfăşurate, după câteva generaţii, populaţiile de dimensiuni mici conţin indivizi mai buni (ca număr şi calitate), faţă de cele cu număr mare de indivizi.

Tabela 5 rezumă soluţiile obţinute pentru reprezentarea solţiei bazată pe entităţi şi obiectivele atinse de soluţiile studiate. Numărul de obiective realizate este calculat folosind refactorizările prezente între genele soluţiilor analizate.

Reprezentarea soluţiei	**Valoarea** α	**Cel mai bun crom.** (dim. pop./ nr. gen.)	**Cel mai bun fitness**	**Timp de execuţie** (secunde)	**Numărul de obiective realizate (%)**				
					Atribute *(1a)*	**Metode (2)**			**Ierarhia de clase** *(3a)*
						(2a)	*(2b)*	*(2c)*	
Bazată pe entităţi	0.3	100c/200g	0.25272	64secs	100	50	50	50/100	100
	0.5	20c/200g	0.3562	14secs	100	50	50	100	100
	0.7	50c/50g	0.45757	9secs	100	50	0	50/100	100

Tabela 5: Cei mai buni indivizi obţinutţi pentru pentru reprezentarea soluţiei bazată pe entităţi, pentru diferite valori ale parametrului α: 0.3, 0.5 şi 0.7

6.9 Analiza soluţiilor pentru MORSSP şi MORSgSP

MORSgSP reprezintă un caz particular al MORSSP, soluţiile evolutive fiind studiate pentru aceste probleme de Secţiunile 6.3 şi 6.7. Prima problemă studiază identificarea unei singure refactorizări care modifică o entitate soft, satisfăcând constrângerile impuse, iar cea de a doua identifică o mulţime de refactorizări posibile pentru fiecare entitate soft.

Cel mai bun individ obţinut în experimentele rulate pentru MORSgSP, a fost pentru o populaţie de 20 cromozomi, după o evoluţie de 200 generaţii. Acesta a fost transpus în reprezentarea soluţiei bazată pe refactorizări pentru MORSSP. Individul astfel obţinut are aceeaşi valoare a funcţiei de fitness ca în forma originală (0.3562). Cel mai bun cromozom înregistrat pentru MORSSP, a fost obţinut pentru o populaţie de 20 indivizi, după o evoluţie de 200 generaţii. Dar acesta nu se poate transpune în reprezentarea soluţiei prezentată în Secţiunea 6.7, existând mai multe refactorizări recomandate pentru fiecare entitate soft.

Mai întâi, o refactorizare poate fi aplicată mai multor entităţi soft, aşa cum r_6 (refactorizarea *AddParameter*) care este aplicată lui m_8 (metoda `print`) din c_3 (clasa `PrintServer`) şi m_{11} (metoda `save`) din c_4 (clasa `FileServer`). Apoi, r_1 (refactorizarea *RenameMethod*) este aplicată apoi pentru aceleaşi metode penru a evidenţia caracterul polimorfic al noii metode denumite `process`. Aceasta înseamnă că există cel puţin două refactorizări care trebuie aplicate metodelor amintite aici (`print` şi `save`). Astfel, transformarea multiplă a enităţilor soft nu poate fi codificată de reprezentarea soluţiei propusă în Secţiunea 6.7.

Algoritmul *RSgSGAEnt* (vezi Subsecţiunea 6.8.1) permite ascunderea informaţiei prin sugerarea încapsulării atributelor. Dar reprezentarea soluţiei nu permite aplicarea a mai mult de o refactorizare fiecărei entităţi soft. Din aceasta rezultă imposibilitatea de a aplica refactorizări relevante entităţilor soft [CC11].

Pentru soluţia propusă de algoritmul *RSSGARef* cu $\alpha = 0.5$, valoarea funcţiei de fitness pentru cel mai bun cromozom (0.4793) este mai bună decât a abordării prezentate în Secţiunea 6.8, aceasta sugerând aplicarea mai multor refactorizări unei singure entităţi soft.

6.10 Concluzii şi direcţii de cercetare viitoare

Acest capitol a studiat abordarea evolutivă pentru rezolvarea MORSSP şi MORSgSP. S-a propus un algoritm genetic adaptat pentru a gestiona funcţia ponderată care să combine cele două obiective conflictuale studiate, minimizarea costului refactorizării şi maximizarea impactului refactorizării asupra entităţilor afectate. S-au analizat două reprezentări ale soluţiei evolutive, rezultatele obţinute în urma unor experimente fiind discutate şi comparate.

Capitolul de faţă se bazează pe articolele [CCV09a, CCV09b, CC09c, CC09b, CC09a, CC09e, CC09d, CC10a, CC10d, CC11]. Printre direcţiile viitoare de cercetare se regăsesc: studierea operatorilor de încrucişare adaptaţi problemei şi studierea frontului Pareto.

7 Concluzii şi direcţii viitoare de cercetare

Scopul prezentei teze este să susţină ideea că tehnicile de refactorizare joacă un rol important în ingineria softului, având avantajul unui domeniu de cercetare activ. Scopul şi obiectivele cercetării desfăşurate au fost îndeplinite de această teză.

Principalele contribuţii ale acestei teze ţintesc trei dintre cei şase paşi de aplicare a unui proces de refactorizare complet: alegerea refactorizărilor potrivite pentru a fi aplicate, evalurea impactului refactorizării asupra atributelor calităţii produselor soft şi păstrarea consistenţei dintre codul refactorizat şi celelalte aspecte ale procesului de dezvoltare a softului. Pentru fiecare aspect discutat sunt prezentate noi direcţii de cercetare.

Selectarea refactorizărilor

Teza abordează aspectul selectării refactorizărilor adecvate în diferite situaţii. Alegerea refactorizărilor potrivite pentru produse soft de dimensiuni diferite este o problemă de cercetare stimulatoare. Dependenţele dintre entităţile soft şi dintre refactorizări sunt elementele de bază care dirijează cercetarea în acest domeniu.

Câteva probleme de selectare multicriterială a refactorizărilor sunt investigate la nivel formal, ca: selectarea unei mulţimi de refactorizări, selectarea unei refactorizări, selectarea unei secvenţe de refactorizări şi construirea unui plan de refactorizare. Formulările multicriteriale folosesc obiective conflictuale, ca: minimizarea costului refactorizării şi maximizarea impactului refactorizărilor asupra entităţilor soft. S-au studiat aspecte legate de procesul de construire a planurilor de refactorizare folosind decizii de management ale proiectelor soft.

Pentru *Problema Selectării Unei Mulţimi de Refactorizări* şi cazul particular al acesteia, *Problema Selectării Unei Refactorizări*, au fost investigate soluţii evolutive. Au fost propuşi algorimi genetici pentru care s-au studiat reprezentări ale soluţiilor bazate de refactorizări şi entităţi. Pentru a compara diferite rezultate ale experimentelor derulate, a fost propusă o strategie de evaluare orientată pe obiective, pentru evaluarea refactorizărilor selectate. Algoritmul genetic propus este bazat pe modelul evolutiv steady-state împreună cu o metodă de agregare ponderată a obiectivelor de optimizat şi operatori genetici adaptaţi.

Direcţii de cercetare viitoare.

Pentru diferite probleme de selectare a refactorizărilor formalizate în cadrul acestei teze, există câteva direcţii în care se poate continua cercetarea. Un aspect important al problemei de construire a planurilor de refactorizare este analizarea direcţiei de compunere paralelă a secvenţelor de refactorizări. Este necesară introducerea de notaţii noi ca punctele de multi-joncţiune care să permită multi-conectarea între secvenţele de refactorizări. De asemenea, este necesar un studiu aprofundat al punctelor de joncţiune, care reprezintă puncte de intersecţie ale secvenţelor de refactorizări.

Folosind diferite criterii în managementul deciziilor din procesul de construire a planurilor de refactorizare, se pot construi diverse strategii. Astfel, se pot enunţa şi descrie principii de construire a planurilor de refactorizare. Cercetarea în configurarea planurilor de refactorizare poate presupune şi descrierea unui model care să evidenţieze diferite scenarii, relaţii şi metode de construcţie pentru diferite secvenţe de refactorizări.

În domeniul selectării refactorizărilor se poate aprofunda studiul asupra unor operatori de încrucişare adaptaţi problemei studiate. Frontul Pareto poate fi studiat în continuare pentru a analiza cei mai buni cromozomi, în contextul existenţei unor obiective conflictuale de optimizat. Alte aspecte care pot fi investigate sunt legate de diferite experimente care să aibă la bază produse soft din lumea reală. Mai mult, diferite abordări ale soluţiilor pentru *MORSqSP* şi *MORPBP* trebuie analizate.

Construirea unui instrument care să permită colectarea datelor de intrare pentru produsele soft analizate, reprezintă un pas important în domeniul ingineriei soft bazată pe căutare ce urmează a fi dezvoltat.

Formalizarea şi evaluarea impactului refactorizării

Impactul refactorizării asupra reprezentării structurii interne ca AST a fost investigat de această teză. Astfel, impactul refactorizării este exprimat prin numărul de vârfuri şi arce afectat în cadrul AST, ca urmare a aplicării refactorizărilor. Descrierea bazată pe impact a fost aplicată asupra câtorva refactorizări relevante: *MoveMethod*, *MoveField*, *ExtractClass* şi *InlineClass*.

Evaluarea impactului refactorizării asupra calităţii interne a programelor poate fi folosită ca indicator al necesităţii de transformare prin refactorizare. Metricle soft au devenit un instrument esenţial în ingineria softului. Acestea sunt folosite pentru a evalua calitatea şi complexitatea produselor soft, dar şi pentru înţelegerea şi furnizarea unor indicii legate de aspectele sensibile ale produselor soft.

S-a oferit o descriere formală a unei mulţimi consistente de metrici soft pentru care s-au folosit notaţii formale introduse anterior. Metricile soft studiate au fost grupate în patru categorii: cuplare, coeziune, complexitate şi abstractizare.

S-a propus o tehnică de analiză a impactului refactorizărilor asupra calităţi interne a programelor exprimată prin metrici soft. Strategia propusă constă în cinci paşi care arată modificarea valorii metricii soft în domeniul de valori al acesteia după aplicarea refactorizării. Aceasta investighează o mulţime de obiective urmărite de către dezvoltator, lista de categorii de impact, regulile de evaluare aplicate pentru a obţine impactul final de refactorizare asupra calităţii interne prin metrici soft.

Clasificarea impactului refactorizării bazată pe intenţia dezvoltatorului furnizează un răspuns acestuia înainte de aplicarea propriu-zisă a refactorizării. Astfel, acesţia sunt capabili să estimeze modificarea apărută asupra calităţii prin refactorizare. Au fost identificate şi două limitări ale abordării propuse pentru care s-au oferit soluţii imediate.

Direcţii de cercetare viitoare.

Cercetarea s-a concentrat asupra impactului refactorizării, iar evaluarea este dirijată în viitor de câteva aspecte. Se poate construi un catalog care să conţină impactul refactorizărilor de diferite categorii asupra structurii interne a programelor reprezentată ca AST. Acest catalog poate fi utilizat ca punct de plecare pentru a construi strategii de refactorizare bazate pe diferite atribute ale calităţii interne.

Diferite refactorizări compuse pot fi studiate în viitor pentru a identifica refactorizările primitive. O asemenea analiză poate evidenţia anumite refactorizări primitive care sunt parte a unor refactorizări complexe şi care anluează impactul altor refactorizări deja aplicate. Limitările reprezentării curente ca AST în descrierea unor refactorizări complexe poate fi eliminată prin extinderea cu notaţii corespunzătoare.

Analiza bazată pe impactul refactorizării necesită validare suplimentară, urmând abordările de validare la nivel teoretic şi practic. Astfel, este necesară analiza produselor soft reale pentru a verifica impactul refactorizării asupra unei mulţimi de metrici soft relevante.

Un alt pas în acest domeniu de cercetare este construirea unor instrumente care să integreze caracteristicile analizei propuse. O abordare de validare care să folosească elementele introduse este un pas imediat ce urmează a fi realizat. De asemenea, va fi abordată compararea cu instrumente de similare. Automatizarea procesului de analiză poate fi legată de instrumentul dezvoltat.

Alte aspecte care necesită o analiză detaliată în viitor sunt posibilele limitări ale abordării de analiză, ca modul în care numărul de metrici soft asupra cărora se evaluează impactul refactorizării poate fi redus. Ulterior, se pot construi cataloage cu metrici soft pentru care abordarea propusă nu se poate aplica datorită informaţiilor insuficiente oferite de modelul de prezentare. De asemenea, tipurile de duplicări de cod la nivelul ierarhiilor de clase se pot formaliza.

Păstrarea consistenţei cu alte etape de dezvoltare

Un alt aspect studiat în cadrul tezei este păstrarea consitenţei codului sursă refactorizat cu celelalte etape ale procesului de dezvoltare. Cercetarea din cadrul modelării conceptuale a arătat că există posibilitatea de a integra procesul de refactorizare în etapa de analiză a procedului de dezvoltare a softului.

Pentru a gestiona diferite tipuri de variabiliate la nivelul modelării conceptuale, s-a propus un model bazat pe evoluţia biologică. Au fost recomandate refactorizări specifice pentru a migra între modele conceptuale ontogenice, iar abstractizarea conceptuală şi specializarea conceptuală au fost indicate pentru realiza evoluţia filetică între modelele conceptuale.

Diferite noţiuni, ca modelele conceptuale şi transformările necesare pentru a migra au fost descrise ca refactorizări, abstractizare conceptuală şi specializare conceptuală şi au fost formalizate. Procesele de evoluţie biologică identificate au fost definite formal. În final, cele trei tipuri de variabilitate conceptuală au fost modelate ca procese ontogenice şi filetice.

Direcţii de cercetare viitoare.

Cercetarea în domenul variabilităţii la nivel conceptual se poate desfăşura în câteva direcţii. Efortul necesar pentru transformarea modelelor conceptuale poate fi estimat şi se poate dezvolta un model de evaluare a procesului de migrare între diferite variante, folosind criterii diferite. Evaluarea criteriilor poate urma cele două modalităţi de evaluare a calităţii, folosind atribute interne şi externe ale calităţii variantelor investigate.

O altă direcţie este legată de aplicarea refactorizărilor la transformarea modelelor conceptuale. Ulterior, atributele calităţii externe pot fi evaluate folosind transformările impuse de refactorizări.

Studierea aprofundată a procesului de transformare între variantele de abstractizare horizontală reprezintă un alt aspect de analizat. Modelarea conceptuală bazată pe ontologii pentru a evidenţia migrarea în cadrul variabilităţii de abstractizare horizontală, reprezintă un nou domeniu de cercetare legat de procesul de refactorizare.

Problema simulării funcţionării unei reţele locale, *Problema ierarhiei de clase cu date structurate* şi un extras din *Problema de management a activităţii didactice* sunt folosite ca şi studii de caz pentru a evidenţia diferite aspecte legate de refactorizările aplicate în cadrul cercetării.

Bibliografie

[Bak95] B. Baker. On finding duplication and near-duplication in large software systems. In *Proceeding of the Second Working Conference on Reverse Engineering (WCRE'95), Toronto, Ontario, Canada*, pages 86–95, 1995.

[Bec99] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.

[Boe88] B.W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72, 1988.

[CC90] E.J. Chikofsky and J.H. Cross. Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1):13–17, 1990.

[CC05a] **M.C. Chisăliţă-Creţu**. Efecte ale refactorizării asupra structurii interne a codului. *"Analele Facultăţii", Seria Ştiinţe Economice, Universitatea Creştină "Dimitrie Cantemir" Bucureşti, Facultatea de Ştiinţe Economice Cluj-Napoca, ISSN:1584-5621*, 13(1):214–230, 2005.

[CC05b] **M.C. Chisăliţă-Creţu**. General aspects of refactoring applicability to conceptual models. In *Proceedings of the Symposium "Colocviul Academic Clujean de INFORMATICĂ"(CACI2005)*, pages 99–104, 2005.

[CC07] **M.C. Chisăliţă-Creţu**. Describing low level problems as patterns and solving them via refactorings. *"Studii şi Cercetări Ştiinţifice", Seria Matematică, ISSN: 1224-2519*, 1(17):29–48, 2007.

[CC09a] **M.C. Chisăliţă-Creţu**. The entity refactoring set selection problem - practical experiments for an evolutionary approach. In *Proceedings of the World Congress on Engineering and Computer Science (WCECS2009), October 20-22, 2009, San Francisco, USA*, pages 285–290. Newswood Limited, ISBN: 978-988-17012-6-8, 2009.

[CC09b] **M.C. Chisăliţă-Creţu**. First results of an evolutionary approach for the entity refactoring set selection problem. In *Proceedings of the 4th International Conference "Interdisciplinarity in Engineering" (INTER-ENG 2009), November 12-13, 2009, Târgu Mureş, România*, pages 303–308. ISSN: 1843-780x, 2009.

[CC09c] **M.C. Chisăliţă-Creţu**. A multi-objective approach for entity refactoring set selection problem. In *Proceedings of the 2nd International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2009), August 4-6, 2009, London, UK*, pages 790–795. ISBN: 978-1-4244-4456-4, 2009.

[CC09d] **M.C. Chisăliţă-Creţu**. Search-based software entity refactoring - a new solution representation for the multi-objective evolutionary approach of the entity

set selection refactoring problem. In *Proceedings of the International Scientific and Professional Conference XXII. DidMatTech 2009, September 10-11, 2009, Trnava, Slovakia*, pages 36–41, 2009.

[CC09e] **M.C. Chisăliţă-Creţu**. Solution representation analysis for the evolutionary approach of the entity refactoring set selection problem. In *Proceedings of the 12th International Multiconference "Information Society" (IS2009), October 12th-16th, 2009, Ljubljana, Slovenia*, pages 269–272. Informacijska družba, ISBN: 978-961-264-010-1, 2009.

[CC10a] **M.C. Chisăliţă-Creţu**. An evolutionary approach for the entity refactoring set selection problem. *Journal of Information Technology Review, ISSN: 0976-2922*, pages 107–118, 2010.

[CC10b] **M.C. Chisăliţă-Creţu**. Formalizing the refactoring impact on internal program quality. In *Proceedings of the Symposium "Zilele Academice Clujene" (ZAC2010)*, pages 86–91, 2010.

[CC10c] **M.C. Chisăliţă-Creţu** and A. Mihiş. A model for conceptual modeling evolution. In *The 7th International Conference on Applied Mathematics (ICAM 2010), September 1-4, 2010, Baia-Mare, România*, 2010.

[CC10d] **M.C. Chisăliţă-Creţu**. The optimal refactoring selection problem - a multi-objective evolutionary approach. In *The 5th International Conference on virtual Learning (ICVL 2010), October 29-31, 2010, Târgu Mureş, România*, pages 410–417, 2010.

[CC10e] **M.C. Chisăliţă-Creţu**. A refactoring impact based approach for the internal quality assessment. In *The 7th International Conference on Applied Mathematics (ICAM 2010), September 1-4, 2010, Baia-Mare, România*, 2010.

[CC10f] **M.C. Chisăliţă-Creţu**. The refactoring plan configuration. a formal model. In *The 5th International Conference on virtual Learning (ICVL 2010), October 29-31, 2010, Târgu Mureş, România*, pages 418–424, 2010.

[CC11] **M.C. Chisăliţă-Creţu**. *Advances in Computer Science and Engineering*, chapter The Entity Refactoring Set Selection Problem - A Solution Representation Analysis. IN-TECH, *accepted book chapter*, 2011.

[CCcC06] **M.C. Chisăliţă-Creţu** and C.A. Şerban. Impact on design quality of refactorings on code via metrics. In *Proceedings of the Symposium "Zilele Academice Clujene" (ZAC2006)*, pages 39–44, 2006.

[CCV09a] **M.C. Chisăliţă-Creţu** and A. Vescan. The multi-objective refactoring selection problem. *Studia Universitatis Babes-Bolyai, Series Informatica, ISSN:2065-9601*, Special Issue KEPT-2009: Knowledge Engineering: Principles and Techniques(July 2009):249–253, 2009.

[CCV09b] **M.C. Chisăliţă-Creţu** and A. Vescan. The multi-objective refactoring selection problem. In *Proceedings of the 2nd Internaltional Conference Knowledge Engineering: Principles and Techniques (KEPT2009)*, pages 291–298. Presa Universitară Clujeană, 2009.

[CY79] L. Constantine and E. Yourdon. *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*. Prentice-Hall, 1979.

[DB04] B. Du Bois. Opportunities and challenges in deriving metric impacts from refactoring postconditions. In *In Proceedings of the Fifth International Workshop on Object Oriented Reengineering (WOOR2004), ECOOPworkshop*, 2004.

[DBM03] B. Du Bois and T. Mens. Describing the impact of refactoring on internal program quality. In *In Proceedings of the International Workshop on Evolution of Large-scale Industrial Software Applications (ELISA), ICSM-workshop*, 2003.

[DDN00] S. Demeyer, S. Ducasse, and O. Nierstrasz. Finding refactorings via change metrics. In *In Proceedings of OOPSLA 2000, ACM SIGPLAN Notices*, pages 166–178, 2000.

[Deu89] L.P. Deutsch. Design reuse and frameworks in the smalltalk-80 system. *Software Reusability: Applications and Experience*, II(1):57–72, 1989.

[Fow99] M. Fowler. *Refactoring Improving the Design of Existing Code.* Addison-Wesley, 1999.

[Fow04] M. Fowler. Refactoringmalapropism, 2004.

[FP97] N. Fenton and S. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* 2nd Edition, PWS Publishing Company, 1997.

[FS97] M. Fayad and D. Schmidt. Object-oriented application frameworks. *Communications of the ACM*, 40(10):32–38, 1997.

[GP95] D. Garlan and D. Perry. Introduction to the special issue on software architecture. *IEEE Transactions on Software Engineering*, 21(4):269–274, 1995.

[IEE92] IEEE. *Standard 1061-1992 for a Software Quality Metrics Methodology.* New York: Institute of Electrical and Electronics Engineers, 1992.

[IEE99] IEEE. *Standard IEEE Std 1219-1999 on Software maintenance Volume 2.* IEEE Press, 1999.

[ISO91] ISO/IEC. *9126 Standard, Information technology. Software product evaluation. Quality characteristics and guidelines for their use.* Switzerland: International Organization For Standardization, 1991.

[ISO99] ISO. *Standard 14764 on Software Engineering. Software Maintenance.* ISO/IEC, 1999.

[Joh93] J.H. Johnson. Identifying redundancy in source code using fingerprints. In *Proceeding of the 1993 Conference of the Centre for Advanced Studies Conference (CASCON'93), Toronto, Canada*, pages 171–183, 1993.

[Kd05] Y. Kim and O.L. deWeck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158, 2005.

[Ker04] J. Kerievsky. *Refactoring to Patterns.* Addison-Wesley Professional, 2004.

[KIAF02] Y. Kataoka, T. Imai, H. Andou, and T. Fukaya. A quantitative evaluation of maintainability enhancement by refactoring. In *Proceedings International Conference on Software Maintenance*, pages 576–585, 2002.

[KN01] G. G. Koni-N'Sapu. A scenario based approach for refactoring duplicated code in object oriented systems. Master's thesis, University of Bern, Diploma thesis, 2001.

[LR00] M.M. Lehman and J.F. Ramil. Towards a theory of software evolution and its practical impact. In *Invited Talk, Proceedings of International Symposium on Principles of Software Evolution (ISPSE2000)*, pages 2–11. Press, 2000.

[Mar02] R. Marinescu. *Measurement and Quality in Object-Oriented Design*. PhD thesis, "Politehnica" University of Timisoara, 2002.

[ML02] T. Mens and M. Lanza. A graph-based metamodel for object-oriented software metrics. *Electronic Notes in Theoretical Computer Science*, 72(2):–, 2002.

[MT04] T. Mens and T. Tourwe. A survey of software refactoring. *IEEE Transactions on Software Engineering*, 30(2):126–139, 2004.

[MV98] H. R. Maturana and F.J. Varela. *The Tree of Knowledge: The Biological Roots of Human Understanding*. Shambhala Publiccations,Inc., Boston, MA., USA, 1998.

[MVEDJ05] T. Mens, N. Van Eetvelde, S. Demeyer, and D. Janssens. Formalizing refactorings with graph transformations. *Journal of Software Maintenance and Evolution: Research and Practice*, 17(4):247–276, 2005.

[Opd92] W.F. Opdyke. *Refactoring Object-Oriented Frameworks, PhD thesis*. Department of Computer Science, University of Illinois at Urbana-Champaign, 1992.

[Par79] D. Parnas. Designing software for ease of extension and contraction. *IEEE Transactions on Software Engineering*, 5(2):128–128, 1979.

[Pig97] T.M. Pigoski. *Practical Software Maintenance. Best Practices for Managing Your Software Investment*. John Wiley and Sons, 1997.

[Rob99] D.B. Roberts. *Practical Analysis for Refactoring*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1999.

[Roy70] W.W. Royce. Managing the development of large software systems: concepts and techniques. In *Proc. IEEE WESTCON, IEEE Computer Society Press (August 1970) Reprinted in Proc. International Conf. Software Engineering (ICSE) 1989, ACM Press*, pages 328–338, 1970.

[Som96] I. Sommerville. *Software Engineering*. Addison-Wesley, fifth edition, 1996.

[Swa76] E.B Swanson. The dimensions of maintenance. In *Proceedings of the 16th International Conference on Software Engineering, IEEE Computer Society*, pages 492–497, 1976.

[TK03] L. Tahvildari and K. Kontogiannis. A metric-based approach to enhance design quality through meta-pattern transformations. In *Proceeding of the European Conference on Software Maintenance and Reengineering*, pages 183–192. IEEE Computer Society Press, 2003.

[Ver04] J. Verelst. The influence of the level of abstraction on the evolvability of conceptual models of information systems. In *Proceedings of the International Symposium on Empirical Software Engineering (IIESE04), Los Angeles, IEEE CS Press*, pages 17–26, 2004.