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

Metrici în evaluarea softului

Rezumatul tezei de doctorat

Doctorand: Camelia Şerban
Conducător ştiinţific: Prof. Dr. Militon Frenţiu

2012

MULŢUMIRI

Mulţumesc tuturor celor care au contribuit la finalizarea acestui demers ştiinţific.

Adresez sincere mulţumiri domnului profesor Militon Frenţiu, coordonatorul stiinţific al acestei teze, pentru ajutorul oferit pe parcursul acestor ani.

Mulţumesc, de asemenea, domnului profesor Horia F. Pop pentru oportunităţile de cercetare oferite şi pentru ideile valoroase.

Pasiunea pentru cercetare, încurajările şi multe idei au venit din partea colegei mele Andreea Vescan, căreia îi sunt profund recunoscătoare.

Mulţumesc din tot sufletul parinţilor mei şi surorii mele pentru ajutorul enorm pe care mi l-au oferit. M-au ajutat să merg mai departe când eu incetasem să mai sper că pot. Pe tot parcurs acestor ani, a fost alături de mine soţul meu, căruia îi mulţumesc pentru ajutorul şi încurajările oferite. Dedic această teză fetiţei mele, Emilia, cu toată dragostea mea.

Mulţumesc tuturor colegilor din Departamentul de Informatică care m-au ajutat şi cu care am colaborat.

Cuprins

Cuprinsul tezei de doctorat

Lista publicaţiilor autorului

- [Ser11] **C. Serban**. God Class Design Flaw Detection In Object Oriented Design. A Case–Study. *Studia Universitas Babes-Bolyai, Seria Informatica*, LVI(4):33-38, 2011. *(MathSciNet)*

- [Ser10] **C. Serban**. A Conceptual Framework for Object-Oriented Design Assessment. *In UKSim 4th European Modelling Symposium on Mathematical Modelling and Computer Simulation*, pages 90-95, 2010. *(IEEE paper)*

- [SVP10b] **C. Serban**, A. Vescan, and H. F. Pop. A Formal Model for Component-Based System Assessment. *In Second international conference on Computational Intelligence, Modelling and Simulation*, pages 261-266, 2010. *(IEEE paper)*

- [SVP10a] **C. Serban**, A. Vescan, and H. F. Pop. A conceptual framework for component-based system metrics definition. *In 9th RoEduNet International Conference*, Sibiu, Romania, pages. 73–78, 2009. *(ISI Proceeding)*

- [Ser09b] **C. Serban**. High coupling detection using fuzzy clustering analysis. *Knowledge Engineering: Principles and Techniques (Post-proceedings of KEPT 2009), International Conference*, Babes-Bolyai University, Presa Universitara Clujeana, pages 258-265, 2009. *(ISI Proceeding)*

- [SVP09] **C. Serban**, A. Vescan, and H.F. Pop. A new component selection algorithm based on metrics and fuzzy clustering analysis. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, LNCS Vol. 5572, pages 621-628, 2009. *(ISI Proceeding)*

- [Ser09a] **C. Serban**, A formal approach for OOD metrics definition. *Proceedings of the First International Conference on Modelling and Development of Intelligent Systems*, Lucian Blaga University Press, pages 262-269, 2009. *(MathSciNet)*

- [Ser09c] **C. Serban**. High coupling detection using fuzzy clustering analysis. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceedings of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 223-226, 2009. *(MathSciNet)*

- [SP08] **C. Serban** and H.F. Pop. Software quality assessment using a fuzzy clustering approach. *Studia Universitas Babes-Bolyai, Seria Informatica,* LIII(2):27-38, 2008. *(MathSciNet)*

- [SVP08] **C. Serban**, A. Vescan, and H.F. Pop. Component selection based on fuzzy clustering analysis. *Creative Mathematics and Informatics*, 17(3):505-510, 2008.

- [SV07b] **C. Serban** and A. Vescan. Metrics for component-based system development. *Creative Mathematics and Informatics*, pages 143-150, 2007. *(MathSciNet)*

- [SV07a] **C. Serban** and A. Vescan. Metrics-based selection of a component assembly. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceedings of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 324-331, 2007. *(MathSciNet)*

- [SC06] **C. Serban** and C. Cretu. Impact on design quality of refactorings on code via metrics. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 39-44, 2006.

- [Ser06] **C. Serban**. Coupling measurement for compiled .Net code. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 21-26, 2006.

- [SM05] **C. Serban** and A. Mihis. Software quality assurance. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 207-212, 2005.

Lista cuvintelor cheie

Metrici soft, proiectarea orientată obiect, dezvoltarea bazată pe componente, evaluarea proiectării orientate obiect, evaluarea sistemelor bazate pe componente, analiza bazată pe clasificarea nuanţată.

1 Introducere

Acestă teză de doctorat este rezultatul cercetării mele în Ingineria Soft, în particular în domeniul Metricilor Soft, cercetare începută în anul 2004.

Creşterea programelor în dimensiune şi complexitate a condus la costuri ridicate de dezvoltare şi productivitate tot mai scazută. Sistemele soft au devenit inflexibile (functionalităţi noi nu pot fi adăugate cu uşurinţă), monolitice (nu există o funcţionalitate bazată pe componente a sistemului) şi greu de întreţinut (încercarea de a aduce noi modifcări determină un lanţ nesfârşit de ajustări în multiple locuri) [Mih03].

Necesitatea obţinerii calităţii în sistemele soft a devenit din ce în ce mai evidentă. Îmbunătăţirea acesteia este posibilă doar prin controlul ei. Dar, aşa cum De Marco afirmă: "nu poţi controla ceea ce nu poţi măsura" [DeM82]. În consecinţă, măsurătoarea soft joacă un rol important în îmbunătăţirea sistemelor în general şi a celor software în particular.

Măsurarea nu este numai utilă, fiind chiar necesară. Ea este esenţială pentru verificarea stadiului proiectelor în curs de dezvoltare şi pentru a afla resursele şi procesele implicate. De asemenea, este dificil să se precizeze dacă un proiect este bun sau rău, în cazul în care masuratorile privind caracteristicile sale (bun, calitativ, etc) nu sunt efectuate [Bar02]. Cu alte cuvinte, proiectele trebuie să fie controlate, mai degrabă decât doar dezvoltate.

În domeniul Ingineriei Soft măsurătorile au devenit esenţiale. Programatorii măsoară diferite caracteristici ale softului pentru a se asigura că cerinţele sunt consistente şi complete, proiectarea este de calitate şi codul este pregătit pentru testare. Managerii proiectelor fac în permanenţă măsurători ale procesului şi produsului pentru a putea determina dacă acesta poate fi livrat. Clienţii verifică dacă produsul final satisface cerinţele şi standardele de calitate. Persoanele care întreţin produsul fac evaluări ale acestuia pentru a decide ce trebuie îmbunătăţit sau adăugat [FP97].

În consecinţă, măsurătorile soft au devenit o preocupare pentru mulţi cercetători. Acestea măsoară diferite aspecte ale softului, jucând astfel un rol important în înţelegerea, controlul şi imbunătăţirea calităţii softului.

Foarte multe măsurători au fost propuse în literatură până acum şi altele noi continuă să fie definite. Totuşi, aceste măsurători ridică probleme majore. Câteva dintre acestea, abordate în lucrare, sunt discutate în continuare.

Adesea *definiţiile metricilor* sunt *incomplete*, *ambigue* şi *deschise diferitelor interpretări* [BDW99, Mar97, MP08]. De exemplu, definiţia metricii NOM dată de Li şi Henry [LH93b] este următoarea: "Numărul de metode locale ale unei clase". Problema este aceea că autorii nu explică termenul de "metodă locală". Nu se face referire la metodele moştenite, nici la metodele de clasă sau cele redefinite. De asemenea, ne punem intrebarea: este luată în considerare vizibilitatea unei metode? Deşi răspunsuri la aceste intrebări nu au fost găsite, autorii validează această metrică ca fiind un bun predictor în ceea ce priveşte efortul de menţinere [Rei01].

Dincolo de problema referitoare la definiţiile metricilor, este cea cu privire la *interpretarea rezultatelor măsurătorilor*. În majoritatea cazurilor nu există modele de interpretare sau sunt foarte empirice, stânjenind astfel aplicabilitatea rezultatelor măsurătorilor. Un alt aspect cu privire la interpretarea rezultatelor obţinute în urma măsurătorilor rezidă în *dificultatea stabilirii valorilor de prag pentru metricile utilizate* [Mar02].

Aşadar, care sunt valorile de prag "corecte" şi cum se stabilesc acestea.

Deoarece metricile sunt întâlnite în toate domeniile Ingineriei Soft, cercetarea noastră va fi limitată la aplicarea acestora în domeniul Proiectării Orientate Obiect (Object Oriented Design - OOD) şi în cel al Dezvoltării Bazate pe Componente (Component Based Development - CBD).

Scopul acestei lucrări este acela de a investiga modul în care metricile pot fi aplicate în evaluarea sistemelor soft, în particular în evaluarea proiectării orientate obiect şi a sistemelor bazate pe componente. De asemenea avem ca scop investigarea unor metode şi tehnici care să ofere o interpretare relevantă a rezultatelor măsurătorilor.

Teza este focalizată pe activitatea de evaluare a softului bazată pe metrici. Ea este structurată pe şapte capitole (un capitol introductiv, unul cu noţiuni fundamentale, patru capitole de contribuţii şi unul de concluzii) şi conţine 132 de referinţe bibliografice.

Capitolul 1 descrie contextul, motivaţia şi scopul tezei, enumeră contribuţiile aduse de autor precum şi modalitatea de diseminare a lor şi prezintă structura generală a lucrării.

Capitolul 2 oferă o introducere în domeniile referite în cadrul tezei. Aceasta include conceptele fundamentale ale proiectării orientate obiect şi a dezvoltarii bazate pe componente, conceptele cheie ale teoriei măsurătorilor, o scurtă trecere în revistă a abordărilor existente în literatură cu privire la utilizarea măsurătorilor soft şi asigurarea calităţii în domeniile OOD şi CBD. De asemenea este prezentată pe scurt metoda de analiză bazată pe clasificarea nuanţată, metodă folosită la interpretarea rezultatelor măsurătorilor. După enumerarea limitărilor abordărilor actuale sunt stabilite obiectivele acestei lucrări.

Capitolul 3 propune un cadru conceptual pentru evaluarea orientată obiect. Cadrul defineşte patru nivele de abstractizare: 1. Modelul proiectării orientate obiect – defineşte într-o manieră formală domeniul evaluării, specificând elementele care sunt evaluate, proprietăţile acestora precum şi relaţiile dintre ele; 2. Definiţii formale ale metricilor pentru OOD – cuprinde o bibliotecă de definiţii formale ale metricilor privind proiectarea orientată obiect; 3. Specificarea obiectivelor evaluării – specifică într-o manieră formală obiectivele evaluării folosind o abordare bazată pe metrici; 4. Analiza rezultatelor evaluării – interpretarea rezultatelor măsurătorilor este realizată folosind metoda de analiză bazată pe clasificarea nuanţată.

Capitolul 4 evaluează metodele şi tehnicile introduse în Capitolul 3 printr-un studiu de caz relevant. Acesta include o comparaţie cu abordările existente bazate pe strategii de detecţie şi care utilizează valori prag pentru metrici. Capitolul prezintă de asemenea unealta soft dezvoltată de noi cu scopul de a automatiza procesul de evaluare al proiectării orientate obiect.

Capitolul 5 propune un cadru de evaluare pentru sistemele bazate pe componente. Acest cadru delimitează problema evaluării componentelor, oferind contextul teoretic pentru definiţiile formale ale metricilor şi pentru specificarea obiectivelor evaluării. Cadrul propus este similar cu cel prezentat în Capitolul 3, dar are alt nivel de granularitate, acela de componentă.

Capitolul 6 propune o evaluare a ansamblului de componente. Este studiată cuplarea sistemului, deoarece aceasta surprinde interacţiunile dintre componente şi este în strânsă legătură cu atributele de calitate. Sunt selectate metricile care cuantifică această interacţiune dintre componente şi sunt propuse câteva noi, studiind în acelaşi

timp şi influenţa valorilor acestor metrici asupra atributelor de calitate. De asemenea, evaluarea propusă oferă suport în selectarea soluţiei optime dintr-o mulţime de configuraţii posibile.

Capitolul 7 încheie lucrarea, oferind o imagine de ansamblu asupra tuturor contribuţiilor raportate în cadrul acesteia.

Contribuţiile originale introduse în această teză sunt prezentate in Capitolele 3, 4, 5, 6 şi sunt următoarele:

Contribuţii la evaluarea proiectării orientate obiect

- O nouă formalizare pentru modelul proiectării orientate obiect [Ser09a] (Secţiunea 3.1). Această formalizare defineşte contextul teoretic al evaluării orientate obiect, context folosit apoi pentru definiţiile metricilor, enunţarea obiectivelor evaluării şi interpretarea rezultatelor măsurătorilor.

- O noua bibliotecă de definiţii formale ale metricilor pentru proiectarea orientată obiect [Ser09a, Ser10] (Secţiunea 3.2). Metricile sunt definite într-o manieră formală, utilizând contextul descris de modelul proiectării orientate obiect. Definiţiile sunt exprimate în termenii algebrici ai mulţimilor şi relaţiilor.

- Un nou formalism pentru specificarea obiectivelor evaluării (Secţiunea 3.3). Obiectivele evaluării sunt specificate utilizând o abordare bazată pe metrici.

- O nouă metodă pentru interpretarea rezultatelor măsurătorilor bazată pe tehnicile de clasificare nuanţate [SP08] (Secţiunea 3.4). Metoda de analiză bazată pe clasificarea nuanţată este utilizată cu scopul de a contracara limitările abordărilor existente care utilizează valori prag pentru metrici.

- O nouă metrică referitoare la distribuţia entropiei carenţei de proiectare [Ser09b, Ser09c] (Secţiunea 3.4.2).

- Un cadru conceptual complet pentru evaluarea proiectării orientate obiect [Ser10] (Capitolul 3).

- Un studiu de caz nou cu scopul de a valida experimental motodologia propusă cu privire la evaluarea proiectării orientate obiect [Ser11, Ser09b, SP08] (Capitolul 4). Acest studiu de caz include de asemenea şi o comparaţie cu abordările existente în literatură, abordări bazate în special pe utilizarea valorilor de prag pentru metrici.

Contribuţii la evaluarea sistemelor bazate pe componente

- O nouă metodă pentru problema selectării componentelor bazată pe utilizarea metricilor şi a metodei de clasificare nuanţată [SVP08] (Secţiunea 5.2).

- Un nou algoritm de selecţie a componentelor bazat pe metrici şi clasificarea nuanţată [SVP09] (Secţiunea 5.3).

- O nouă formalizare a definiţiilor metricilor pentru sistemele bazate pe componente [SVP10a] (Secţiunea 5.4.3).
- O nouă metodă pentru interpretarea rezultatelor măsurătorilor utilizând metoda de analiză bazată pe clasificarea nuanţată [SVP10b] (Secţiunea 5.4.4).
- Un cadru conceptual complet pentru evaluarea sistemelor bazate pe componente [SVP10b] (Secţiunea 5.4). Acest cadru delimitează problema evaluării componentelor, definind un model de evaluare a sistemelor bazate pe componente, model ce ofera contextul teoretic pentru definiţiile formale ale metricilor şi pentru specificarea obiectivelor evaluării.
- O nouă abordare formală a unui ansablu de componente văzut ca un graf [SV07b] (Secţiunea 6.1.2).
- O mulţime de metrici noi ce cuantifică atributele de calitate referitoare la un ansablu de componente [SV07b] (Secţiunea 6.1.3).
- O mulţime de metrici noi pentru selectarea celei mai bune soluţii dintr-o mulţime de configuraţii posibile [SV07a] (Secţiunea 6.2.2).
- Un studiu cu privire la influenţa valorilor metricilor asupra atributelor de calitate [SV07a] (Secţiunea 6.2.3).

2 Contextul lucrării

Această lucrare abordează problema aplicării măsurătorilor în două domenii importante ale Ingineriei Soft: cel al proiectării orientate obiect şi al sistemelor bazate pe componente. Astfel, capitolul curent prezintă conceptele fundamentale ale domeniilor menţionate anterior, particularizând modul în care măsurătorile sunt aplicate în aceste domenii.

2.1 Măsurători soft

Cadrul conceptual pentru măsurători propus de Fenton [Fen95] este acceptat tot mai mult de cercetătorii în ingineria soft, oferind suportul teoretic adecvat în această cercetare.

2.2 Metrici în evaluarea proiectării orientate obiect

Datorită complexităţii proiectării sistemelor soft, evaluarea ei a devenit o activitate de lungă durată. Astfel, sunt necesare metode şi tehnici pentru a automatiza procesul de evaluare. Metricile soft sunt o soluţie alternativă, fiind totodată un mijloc de a cuantifica aspectele considerate importante pentru evaluare.

2.2.1 Proiectarea orientată obiect

Pentru a îndeplini atributele de calitate, o aplicaţie dezvoltată conform paradigmei orientate obiect trebuie să respecte anumite *principii*, *euristici* şi *şabloane de proiectare*. Sistemele orientate obiect care nu respectă aceste reguli ce asigură o bună proiectare întâmpină probleme care stânjenesc evoluţia lor.

Riel [Rie96] propune o mulţime de euristici de proiectare, argumentând cu câteva carenţe de proiectare care rezultă dacă aceste reguli sunt încălcate. Fowler [FBB+99] completează cu câteva afirmaţii ce exprimă deviaţii de la o bună proiectare şi consecinţele acestor deviaţii. Martin [Mar] dezbate principiile de proiectare de bază argumentând faptul că încălcarea acestor principii conduce la o proiectare defectuoasă.

2.2.2 Metrici pentru proiectarea orientată obiect

Există foarte multe măsurători pentru proiectarea orientată obiect definite în literatură. Dintre acestea, metricile propuse de Abreu [Abr93, AR94], Chidamber şi Kemerer [CK94], Li şi Henry [LH93a], Lorenz şi Kidd [LK94] sunt cele mai utilizate. Marinescu [Mar02] a clasificat aceste metrici după patru caracteristici esenţiale orientării obiect: *cuplare*, *moştenire*, *coeziune* şi *complexitatea structurală*.

Mai mulţi autori [Mar02, MP08, Rei01, BBA02, MSL06, BDW99, WBD98] au discutat problemele întâlnite atunci când abordarea bazată pe măsurători este aplicată în evaluarea softului. Cele mai importante dintre aceste probleme sunt legate de *definiţiile imprecise ale metricilor* şi *interpretarea rezultatelor măsurătorilor*.

2.3 Metrici în dezvoltarea bazată pe componente

În prezent, dezvoltarea softului bazată pe componente (CBSD) a fost adoptată în industrie ca o paradigmă nouă de dezvoltare. Aceasta propune proiectarea şi construcţia

unui sistem soft prin selectarea şi integrarea componentelor dezvoltate anterior şi apoi asamblarea acestora pentru a obţine funcţionalitatea dorită.

2.3.1 Specificarea şi definiţia componentei soft

În literatura de specialitate [Spa00, Mic, DW98, Szy98] au fost propuse mai multe definiţii ale componentei soft. Deşi aceste definiţii conţin disimilarităţi, toate afirmă că o componentă este un modul soft ce oferă funcţionalităţi prin intermediul interfeţelor.

2.3.2 Metrici în dezvoltarea bazată pe componente

Atunci când se construieşte un sistem soft bazat pe componente, de obicei există mai multe componente care oferă aceeaşi funcţionalitate. Pentru a diferenţia componentele ce oferă funcţionalităţi similare, informaţiile privind atributele de calitate sunt extrem de importante. Ca un rezultat în această direcţie, metricile soft sunt foarte utile, fiind un mijloc de a cuantifica aspectele considerate importante pentru entităţile evaluate(componentele soft). Evaluarea are ca scop obţinerea unei soluţii de calitate cât mai bună. Pe de altă parte, metricile soft pot fi utilizate şi în evaluarea unei soluţii sistem pentru a deferenţia între mai multe soluţii alternative.

Metricile tradiţionale nu pot fi aplicate la dezvoltarea bazată pe componente. Aceasta se datorează în primul rând proprietăţii de "cutie neagră" a componentei [GG03a]. Gill [GG03a] propune câteva aspecte care ar trebui luate în considerare atunci când definim metrici pentru dezvoltarea bazată pe componente.

2.4 Analiza nuanţată în interpretarea rezultatelor măsurătorilor

Aşa cum am menţionat anterior metricile soft ridică câteva probleme, cea mai importantă fiind legată de interpretarea rezultatelor măsurătorilor. Aceasta la rândul ei se datorează şi faptului că valorile prag pentru metrici sunt greu de stabilit. Pentru a contracara limitările abordărilor existente referitoare la aceste probleme, propunem metoda de analiză bazată pe clasificarea nuanţată.

3 Un cadru conceptual pentru evaluarea proiectării orientate obiect

Capitolul curent este focalizat pe dezvoltarea unei metodologii cu privire la evaluarea cantitativă a proiectării orientate obiect. Medologia propusă se bazează pe analiza statică a codului sursă şi este descrisă de un cadru conceptual definit de patru nivele de abstractizare: Modelul proiectării orientate obiect, Definiţii formale ale metricilor pentru proiectarea orientată obiect, Specificarea obiectivelor evaluării, Analiza rezultatelor măsurătorilor. Aceste elemente sunt descrise în secţiunile următoare.

3.1 Un model pentru proiectarea orientată obiect

Modelul proiectării orientate obiect propus de Marinescu [Mar02] este formalizat în abordarea noastră utilizând noţiuni algebrice din teoria mulţimilor şi a relaţiilor.

Definiţia 3.1.1 ([Ser09a, Ser10]) *Tripletul* $D(S) = (E, Prop(E), Rel(E))$ *defineşte un model pentru proiectarea orientată obiect corespunzătoare unui sistem soft* S*, unde:*

- E *reprezintă mulţimea de entităţi de proiectare ale sistemului* S*;*
- $Prop(E)$ *reprezintă proprietăţile elementelor din mulţimea* E*;*
- $Rel(E)$ *reprezintă relaţiile definite între entităţile de proiectare din* E*.*

Elementele $E, Prop(E), Rel(E)$, referite în Definiţia 3.1.1 vor fi prezentate pe scurt în cele ce urmează.

3.1.1 Entităţi de proiectare

Considerăm $E = \{e_1, e_2, ..., e_{noE}\}$ mulţimea entităţilor de proiectare a unui sistem soft S. O entitate e_i, $1 \leq i \leq noE$ poate fi un *pachet*, o *clasă*, o *metodă* a unei clase, un *atribut* a unei clase, un *parametru* al unei metode, o *variabilă locală* declarată în implementarea unei metode sau o *variabilă globală*. Astfel, mulţimea de entităţi de proiectare este definită de următoarea formulă:

$$Design\ Entities = Package(E) \cup Class(E) \cup Meth(E) \cup Var(E), \text{ unde:}$$

$$Var(E) = Attr(E) \cup Param(E) \cup LocVar(E) \cup GVar(E).$$

Notaţiile utilizate în formula de mai sus sunt descrise în detaliu în teză.

3.1.2 Proprietăţile entităţilor de proiectare

Aşa cum am menţionat înainte, al doilea element al modelului nostru este reprezentat de mulţimea proprietăţilor entităţilor de proiectare, notată $Prop(E)$. Deoarece abordarea noastră se referă la şase tipuri de entităţi de proiectare: clasă, metodă, atribut, parametru, variabilă locală, varibilă globală, pentru specificarea proprietăţilor acestor tipuri de entităţi am definit un model. Acest model este descris în teză. Prezentăm specificarea proprietăţilor entităţilor de tip *clasă*.

Definiţia 3.1.4 ([Ser09a, Ser10]) *Cvadruplu* $Prop_{C,Class(E)} = [C, Class(E), Prop_C, PropVal_C]$ *reprezintă specificarea proprietăţilor pentru entităţi de tipul "clasă" unde:*

- $Prop_C = \{Abstractizare, Vizibilitate, Reutilizabilitate\}$;
- $Abstractizare = \{concretă, abstractă, interfaţă\}$;
- $Vizibilitate = \{pachet, interioară, publică\}$;
- $Reutilizabilitate = \{definită\text{-}de\text{-}utilizator, extinsă\text{-}de\text{-}utilizator, bibliotecă\}$.

3.1.3 Relaţii între entităţile de proiectare

În acestă secţiune prezentăm pe scurt tipurile de relaţii definite între entităţile de proiectare. Menţionăm că aceste relaţii se referă doar la interacţiunile directe între entităţi.
Relaţii de moştenire între clase. Considerăm $a, b \in Class(E)$. Există două tipuri de relaţii directe între clase.

- "**a extinde b**", dacă clasa a este o specializare a clasei b (clasa a moşteneşte structura şi comportamentul clasei b);
- "**a implementează b**", dacă clasa a este o implementare a interfeţei b (clasa a implementează comportamentul înterfeţei b).

Relaţia de invocare de metodă. Metricile de cuplare pentru o clasă c pot fi definite dacă cunoaştem metodele apelate de orice metodă $m \in Meth(E)$ şi variabilele referite de orice metodă a clasei c. Definiţia introdusă de Briand [BDW99] pentru apeluri de metodă este adaptată cadrului nostru din lucrare.

Relaţia de referire de atribut. Atributele pot fi referite de metode. Deoarece referinţele de atribute nu sunt determinate dinamic, este suficient să considerăm tipul static al obiectului pentru care atributul este referit.

3.2 Definiţii formale ale metricilor orientate obiect

Oferim un exemplu de aplicare a modelului propus pentru definiţiile formale ale metricilor, prezentând definiţia metricii Method Coupling – MC [RL92].

Definiţia 3.2.1 ([RL92]) ***Method Coupling – MC***
Definiţia informală. *Metrica MC este definită ca numărul de referinţe care nu sunt locale dintr-o metodă.*
Definiţia formală. $\forall c \in Class(E),\ \forall m_0 \in Meth(c)$*:*

$$MC(c, m_0) = card(A \cup B),\ unde:$$

$$A = \{m \in Meth(E) - Meth(c) | m_0\ call\ m\})$$

$$B = \{a \in ARefRel(m) | a \notin Attr(c)\}$$

Comentarii. *O referinţă care nu e locală este definită de autori ca acea referinţă de metodă sau varibilă care nu este definită în clasa metodei măsurate.*

3.3 Problema evaluării proiectării. Stabilirea obiectivelor

Evaluarea proiectării orientate obiect are ca scop verificarea dacă sistemul obţinut îndeplineşte anumiţi factori de calitate, de exemplu uşurinţa de întreţinere, extensibilitatea, scalabilitatea şi reutilizabilitatea. Fenton [Fen94] afirma că o structură internă bună ar trebui sa ofere o bună calitate externă. Astfel, obiectivele evaluării se reduc la verficarea dacă proiectarea sistemului este în conformitate cu principiile şi euristicile unei bune proiectări, acestea fiind în strânsă legătură cu atributele interne de calitate precum cuplare, coeziune, complexitate şi abstractizare a datelor.

Comunitatea cercetătorilor a fost interesată de stabilirea unei legături între principiile unei bune proiectări şi carenţele de proiectare. Ei au vrut sa stabilească pentru o anumită carenţă de proiectare care sunt principiile ce nu au fost respectate în proiectarea sistemului şi viceversa, care sunt carenţele de proiectare apărute în urma nerespectării unui anumit principiu de proiectare.

Deoarece obiectivul nostru este de realiza o evaluare cantitativă a proiectării orientate obiect, pe lângă stabilirea unei legături între principiile de proiectare şi carenţele de proiectare, urmărim de asemenea relaţionarea acestora cu metricile soft. În consecinţă definim două abordări în stabilirea obiectivelor evaluării:

1. În prima abordare pornim de la o mulţime de principii de proiectare, pentru fiecare dintre acestea vom identifica carenţele de proiectare corespunzătoare iar pentru fiecare carenţă o mulţime de metrici; acestă abordare în stabilirea obiectivelor este denumită Principiu-Carenţă-Metrică;

2. A doua abordare porneşte de la o mulţime de carenţe de proiectare, pentru fiecare identificând o mulţime de principii de proiectare corespunzătoare, iar pentru fiecare principiu o mulţime de metrici; această abordare va fi denunită Carenţă-Principiu-Metrică.

Problema evaluării proiectării

Problema evaluării proiectării, referită în teză "design assessment problem (DAP)" constă în identificarea acelor entităţi de proiectare, denumite entităţi "suspecte" care nu respectă un anumit principiu/euristică/regulă de proiectare sau care sunt afectate de anumite carenţe de proiectare. Notaţiile utilizate în descrierea formală a problemei DAP sunt prezentate în teză.

3.4 Analiza rezultatelor evaluării

Aşa cum am afirmat în capitolul precedent, cea mai dificilă etapă în orice activitate de evaluare este reprezentată de analiza rezultatelor sau altfel spus de interpretarea rezultatelor măsurătorilor.

Pentru interpretarea rezultatelor măsurătorilor, Marinescu introduce un mecanism nou, denumit "strategie de detecţie" care combină diferite metrici, filtrează rezultatele şi asociază această informaţie pentru a detecta probleme în arhitectura proiectării. El defineşte strategia de detecţie ca o expresie cuantificabilă a unei reguli prin care fragmente de proiectare care sunt conforme cu acea regulă pot fi identificate în codul sursă [Mar02].

O limitare a acestei abordări este faptul că problema stabilirii valorilor de prag pentru metrici nu este discutată. Aşadar, care sunt valorile de prag corecte şi cum se stabilesc ele? Pentru a contracara această limitare, propunem o nouă metodă privind interpretarea rezultatelor măsurătorilor, metodă ce are la bază clasificarea nuaţată. Astfel, pornim de la abordarea propusă de Marinescu, utilizând prima parte a strategiei de detecţie şi în schimbul mecanismelor de filtrare şi compozitie propunem analiza nuanţată, scopul principal fiind acela de a elimina problema stabilirii valorilor de prag pentru metrici.

3.4.1 Analiza bazată pe clasificarea nuanţată în interpretarea rezultatelor măsurătorilor

Considerăm problema evaluării proiectări DAP definită în teză. Rezultatele evaluării constau dintr-o mulţime AE_p de entităţi de proiectare care sunt evaluate, împreună cu valorile corespunzătoare ale metricilor selectate $M_p = \{m_1, m_2, ..., m_{noM_p}\}$. Formal, rezultatele evaluării pot fi exprimate după cum urmează:

- $AE_p = \{e_1, e_2, ..., e_n\}$, mulţimea de entităţi de proiectare $AE_p \subseteq AE$,
- $e_i = (e_{i1}, e_{i2}, ..., e_{i(noM_p)})$, valorile corespunzătoare ale metricilor $m_1, m_2, ..., m_{noM_p}$, $m_j \in M_p$, $1 \leq i \leq n$, $1 \leq j \leq noM_p$,
- $(e_{ij})_{i=\overline{1,n};j=\overline{1,noM_p}}$, *matricea rezultatelor evaluării.*

În funcţie de valorile metricilor vom selecta din mulţimea AE_p acele entităţi considerate "suspecte" (entităţi care nu respectă un anumit principiu de proiectare p sau care sunt afectate de o anumită carenţă de proiectare p). Metoda de analiză bazată pe clasificarea nuanţată este utilizată pentru a partiţiona entităţile în clusteri.

Definiţia 3.4.1 ([Ser10]) *O mulţime $U_{AE_p,M_p} = \{U_1, U_2, ..., U_c\}$ se numeşte **partiţie nuanţată** a entităţilor de proiectare $AE_p = \{e_1, e_2, ..., e_n\}$, entităţi caracterizate de valorile metricilor $M_p = \{m_1, m_2, ..., m_{noM_p}\}$ ddacă:*

- $U_i = (u_{i1}, u_{i2}, ..., u_{in})$, $1 \leq i \leq c$;
- $u_{ij} \in [0..1]$, $1 \leq i \leq c$, $1 \leq j \leq n$, u_{ij} *reprezintă gradul de apartenenţă al entităţii* e_j *la clusterul* i;
- $\sum_{i=1}^{c} u_{ij} = 1$, $1 \leq j \leq n$.

Algoritmul genetic de clasificare nuanţată [Bez81] folosit pentru obţinerea acestei partiţii, numit Fuzzy c-means are dezavantajul că numărul de clusteri este o dată de intrare. Din acest motiv, vom considera algoritmul Fuzzy Divisive Hierarchic Clustering (FDHC) [Dum88] pentru determinarea partiţiei optime. Partiţia obţinută în urma aplicării acestui algoritm este definită mai jos.

Definiţia 3.4.2 ([Ser10]) *O mulţime $BTFP_{AE_p,M_p} = \{N_1, N_2, ..., N_l\}$ se numeşte **arbore binar de partiţie nuanţată** a entităţilor de proiectare $AE_p = \{e_1, e_2, ..., e_n\}$, entităţi caracterizate de valorile metricilor $M_p = \{m_1, m_2, ..., m_{noM_p}\}$ ddacă:*

- $\forall i \in \{1,2,...,l\}$, $N_i = (f_i, U_i)$, *unde* f_i *este tatăl clusterului* i, $f_i \in \{0...l\}$ *şi* $U_i = (u_{i1}, u_{i2}, ...u_{in})$, u_{ij} *reprezintă gradul de apartenenţă al entităţii* e_j *la clusterul* i, $u_{ij} \in [0..1]$, $j \in \{1,2,...,n\}$;

- $\exists!\ i \in \{1,2,...,l\}$ *astfel încât* $f_i = 0 \wedge u_{ij} = 1, \forall j \in \{1,2,...,n\}$, N_i - *nodul rădăcină;*

- $\forall i \in \{1...l\}$, $f_i \neq 0$, $\exists! j,k$ *astfel încât* $f_i = k \wedge f_j = k \wedge U_i + U_j = U_k$, *unde* $U_i + U_j = (u_{i1} + u_{j1}, u_{i2} + u_{j1}, ..., u_{i(n)} + u_{j(n)})$.

Definiţia 3.4.3 ([Ser10]) *Considerăm un arbore binar de partiţie nuanţată* $BTFP_{AE_p,M_p} = \{N_1, N_2, ..., N_l\}$. *Un nod* $N_i = (f_i, U_i) \in BTFP_{AE_p,M_p}$ *se numeşte* ***nod terminal*** *sau* ***nod frunză*** *ddacă:*

$$\forall j \in \{1,2,...,l\}, j \neq i \text{ and } N_j = (f_j, U_j) \in BTFP_{AE_p,M_p} \Rightarrow f_j \neq i.$$

Definiţia 3.4.4 ([Ser10]) *Considerăm un arbore binar de partiţie nuanţată* $BTFP_{AE_p,M_p}$. *O submulţime* $OFP_{AE,M} = \{G_1, G_2, ..., G_k\} \subset BTFP_{AE_p,M_p}$ *se numeşte* ***partiţie nuanţată optimă*** *a entităţilor de proiectare* AE_p *în raport cu mulţimea de metrici* M_p *ddacă:* G_j *este nod terminal din* $BTFP_{AE_p,M_p}$, $\forall j \in \{1,2,...,k\}$.

În secţiunea următoare vom introduce o nouă metrică, denumită entropia carenţei de proiectare (Design Flaw Entropy (DFE)), metrică ce măsoară dispersia carenţei de proiectare analizată.

3.4.2 Metrica Design Flaw Entropy

Metrica DFE este definită considerând noţiunea de entropie adaptată din teoria informaţiei introdusă de Shannon [SW49]. Pornind de la acest concept mulţi cercetători [EGH02, BDE99, MBA04, KSW95] au definit metrici noi pentru evaluarea produselor soft.

Considerăm o partiţie nuanţată $U_{AE_p,M_p} = \{U_1, U_2, ..., U_c\}$ a entităţilor de proiectare $AE_p = \{e_1, e_2, ..., e_n\}$, entităţi caracterizate de valorile metricilor $M_p = \{m_1, m_2, ..., m_{noM_p}\}$, metrici selectate pentru a cuantifica un anumit principiu de proiectare sau o anumită carenţă de proiectare p.

Definiţia 3.4.5 ([Ser09b]) *Entitatea* $e_j \in AE$, $1 \leq j \leq n$, *are grad de apartenenţă dominant la clusterul* U_i, $1 \leq i \leq c$, *dacă* $u_{ij} = \max\{u_{rj} | r = \overline{1,c}\}$.

Definiţia 3.4.6 ([Ser09b]) *Frecvenţa relativă de apariţie sau de probabilitate a unui cluster* $U_i \in U_{AE_p,M_p}$, *notată* $p(U_i)$ *reprezintă raportul dintre numărul de entităţi din* AE_p *care au grad dominant de apartenenţă la clusterul* U_i *şi numărul total de entităţi din* AE_p.

Vom nota cu $P_{U_{AE_p,M_p}} = \{p(U_1), p(U_2), ..., p(U_c)\}$ distribuţia probabilităţii pe clusteri a partiţiei U_{AE_p,M_p}.

Definiţia 3.4.7 ([Ser09b]) *O măsură a informaţiei(self-information) conţinută în clusterul* $U_i \in U_{AE_p,M_p}$ *este definită ca* $I(U_i) = -\log_2 p(U_i)$.

În contextul definiţiilor şi notaţiilor anterioare, vom prezenta definiţia metricii propuse.

Definiţia 3.4.8 ([Ser09b]) *Entropia carenţei de proiectare (**Design Flaw Entropy (DFE)**) corespunzătoare unei partiţii nuanţate U_{AE_p,M_p} este definită ca o medie a informaţiilor asociate fiecărui cluster $U_i \in U_{AE_p,M_p}$. Formal:*

$$DFE : FP(AE_p, M_p) \rightarrow [0..\infty],\ DFE(U_{AE_p,M_p}) = \sum_{i=1}^{c} p(U_i) \cdot I(U_i)$$

unde $FP(AE_p, M_p)$ este mulţimea tuturor partiţiilor nuanţate corespunzătoare entităţilor de proiectare AE_p, entităţi caracterizate de valorile metricilor $M_p = \{m_1, m_2, ..., m_{noM_p}\}$, metrici selectate pentru a cuantifica un anumit principiu de proiectare sau o anumită carenţă de proiectare p.

Aşa cum am menţionat mai devreme, această metrică măsoară dispersia carenţei de proiectare analizate, oferind în acelaşi timp o analiză mai profundă în interpretarea rezultatelor măsurătorilor.

3.5 Concluzii

În capitolul curent a fost propusă o nouă metodologie pentru evaluarea cantitativă a proiectării orientate obiect. Medologia propusă se bazează pe analiza statică a codului sursă şi este descrisă de un cadru conceptual definit de patru nivele de abstractizare. Au fost abordate două probleme importante cu privire la evaluarea proiectării orientate obiect şi au fost prezentate soluţiile propuse. Contribuţiile originale ale acestui capitol au fost diseminate în articolele [Ser09a, Ser10, Ser09c, Ser09b, SP08].

4 Evaluarea experimentală a modelului propus

Capitolul curent prezintă evaluarea experimentală a modelului propus pentru evaluarea proiectării orientate obiect (Capitolul 3) precum şi unealta soft dezvoltată pentru automatizarea procesului de calculare a metricilor şi de determinare a partiţiilor nuanţate.

4.1 Detecţia carenţelor de proiectare. Studiu de caz

Studiu de caz propus se bazează pe analiza codului sursă al unei aplicaţii orientate obiect denumită log4net [log]. Aceasta conţine 214 clase grupate in 10 pachete.

4.1.1 Identificarea domeniului evaluării

Codul sursă al aplicaţiei *log4net* este analizat cu unealta soft dezvoltată de noi, identificând elementele domeniului evaluării $D(log4net) = (E, Prop(E), Rel(E))$.

4.1.2 Stabilirea obiectivelor evaluării

Obiectivul principal al evaluării este de a identifica acele entităţi de proiectare afectate de carenţele "God Class" [FBB+99] şi "Shotgun Surgery" [FBB+99]. În consecinţă, entităţile evaluate AE se identifică cu mulţimea claselor din sistemul analizat. În cele ce urmează vom identifica celelalte două componente din specificarea obiectivelor evaluării $AO_2(log4net) = (AE, PFG, PMG)$, elementele grafului principii-carenţe precum şi elementele grafului principii-metrici.

Carenţa de proiectare "God Class". Pentru a detecta entităţile de proiectare afectate de această carenţă, Salehie et al. [MSL06] au relaţionat-o cu trei euristici propuse de Riel [Rie96]: *distribuie inteligenţa sistemului uniform*; *atenţie la clase cu comportament non-comunicativ*, *atenţie la clase care accesează datele altor clase*.

Euristicile selectate sunt relaţionate cu următoarele metrici: Weighted Method per Class (WMC) [CK94], Tight Class Cohesion (TCC) [BK95] şi Access to Foreign Data (ATFD) [Mar02].

Analizând definiţiile acestor metrici putem concluziona că o entitate posibil suspectă de această carenţă de proiectare are valor *mari* pentru metricile WMC şi ATFD şi valori *mici* pentru metrica TCC. Este utilizată metoda de analiză bazată pe clasificarea nuanţată pentru a evita abordarea care foloseşte valori prag pentru metrici.

Carenţa de proiectare "Shotgun Surgery". O clasă care este cuplată cu un număr mare de alte clase şi care produce un număr mare de schimbări în sistem poate fi considerată posibil suspectă de carenţa de proiectare "Shotgun Surgery" [FBB+99]. Pe scurt această carenţă de proiectare abordează problema cuplării excesive [Mar02].

Metricile selectate pentru a cuantifica principiile de proiectare asociate cu această carenţă de proiectare sunt: Changing Methods (CM) [Mar02], Weighted Changing Methods (WCM) [Mar02] and Changing Classes (CC) [Mar02].

Pentru a contura mai bine specificarea obiectivelor evaluării, Figura 1 rezumă mulţimile descrise mai sus.

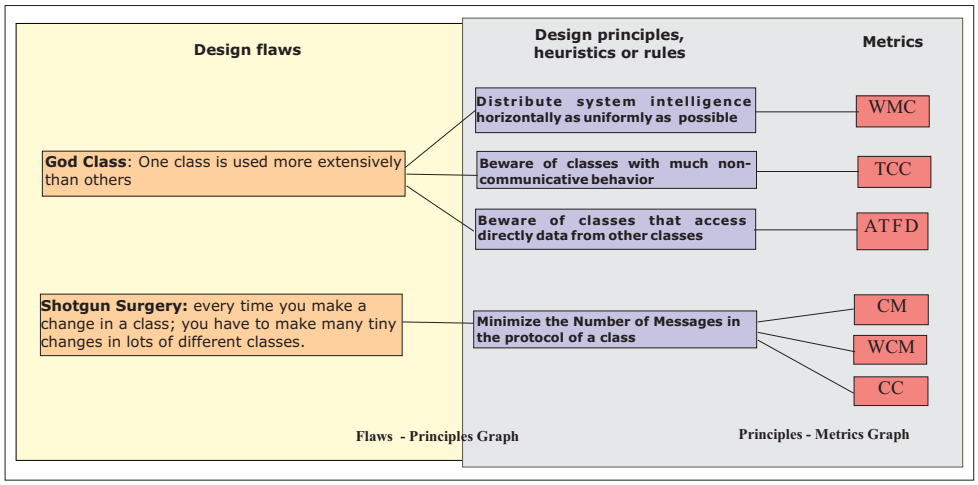

Figura 1: Specificarea obiectivelor evalării pentru studiul de caz propus.

4.1.3 Definiţiile formale ale metricilor selectate

Definiţiile formale ale metricilor selectate pentru studiul de caz propus sunt prezentate în teză.

4.1.4 Determinarea partiţiilor nuanţate. Analiza rezultatelor pentru carenţa de proiectare "God Class"

Al treilea pas în evaluarea propusă constă în aplicarea metodei de analiză bazată pe clasificarea nuanţată cu scopul de a identifica acele entităţi de proiectare afecatate de carenţa de proiectare "God Class". Modul în care vom aplica această metodă este descris în doi paşi după cum urmează:

- Determinarea arborelui binar de partiţie nuanţată utilizând algoritmul ierarhic diviziv de clasificare nuanţată FDHC [Dum88]. Acest arbore este folosit în identificarea partiţiei optime.
- Analiza partiţiilor obţinute pentru a identifica lista entităţilor suspecte.

O comparaţie cu abordarea similară propusă de Marinescu [Mar02], bazată pe strategii de detecţie, este propusă în teză în Secţiunea 4.1.4.3.

4.1.5 Determinarea partiţiilor nuanţate. Analiza rezultatelor pentru carenţa de proiectare "Shotgun Surgery"

Pentru identificarea entităţilor afectate de carenţa de proiectare "Shotgun Surgery" s-a procedat analog cazului carenţei de proiectare "God Class". O comparaţie cu abordarea similară propusă de Marinescu [Mar02], bazată pe strategii de detecţie, este de asemenea propusă în teză în Secţiunea 4.1.5.3.

4.2 Unealta soft dezvoltată

Pentru a reduce timpul necesar pentru evaluarea propusă sunt necesare unelte soft pentru automatizarea procesului. În acest scop, am dezvoltat o aplicaţie denumită Metrics. Arhitectura acestei aplicaţii este descrisă pe scurt în teză.

4.3 Concluzii

În acest capitol am prezentat un nou studiu de caz pentru a valida experimental metodologia propusă cu privire la evaluarea proiectării orientate obiect. Studiu de caz propus abordează problema detecţiei carenţelor de proiectare şi se bazează pe utilizare metricilor şi a tehnicilor de clasificare nuanţată. Contribuţiile originale raportate în acest capitol au fost diseminate în articolul [Ser11].

5 O abordare bazată pe metrici pentru Problema Selectării Componentelor

În acest capitol abordăm problema cunoscută în literatură ca şi Problema Selectării Componentelor (CSP). Scopul nostru este de a selecta o submulţime de componente dintr-o mulţime de componente disponibile, care să îndeplinească cerinţele sistemului. Pentru a selecta cea mai bună componentă dintr-o mulţime de componente care oferă funcţionalităţi similare, informaţiile privind atributele de calitate ar putea fi considerate în procesul de selecţie. Prin urmare, evaluarea componentelor este foarte importantă, iar metricile soft pot fi utilizate în cuantificarea atributelor de calitate, considerate importante pentru sistemul ce urmează a fi construit.

5.1 Problema Selectării Componentelor. Abordare formală

Problema Selectării Componentelor este problema alegerii unei submulţimi de componente dintr-o mulţime disponibilă astfel încât compoziţia lor să satisfacă o mulţime de obiective. Aceste obiective reprezintă mulţimea cerinţelor sistem. O cerinţă este de fapt un serviciu oferit de o componentă.

În cele ce urmează, adoptând notaţiile introduse în [FBR04, Ves08b] vom defini într-o manieră formală Problema Selectării Componentelor.

Considerăm $SR = \{r_1, r_2, ..., r_n\}$ mulţimea de cerinţe sistem şi $CR = \{c_1, c_2, ..., c_m\}$ mulţimea de componente disponibile pentru selecţie. Fiecare componentă $c \in CR$ este specificată prin mulţimea serviciilor oferite $PI_c = \{pi_{c,1}, pi_{c,2}, ..., pi_{c,noPI_c}\}$, mulţimea serviciilor cerute $RI_c = \{ri_{c,1}, ri_{c,2}, ..., ri_{c,noPI_c}\}$ şi mulţimea de dependenţe între serviciile oferite şi cele cerute. Astfel, o componentă $c \in CR$ poate satisface o submulţime de cerinţe din SR.

Scopul este de a determina o submulţime de componente Sol astfel încât fiecărei cerinţe r din mulţimea SR să îi poată fi atribuită o componentă c din Sol, unde r este din PI_c.

5.2 Evaluarea bazată pe metrici a componentelor. Contextul teoretic

Considerăm Problema Selectării Componentelor definită în secţiunea precedentă. Aşa cum am menţionat, selectarea celei mai bune componente dintr-o mulţime ce oferă funcţionalităţi similare necesită informaţii privind atributele de calitate pe care dorim să le deţină sistemul ce se va construi. Prin urmare, evaluarea componentelor este foarte importantă, iar metricile soft pot fi utilizate în cuantificarea atributelor de calitate, considerate importante pentru sistem. Analizând rezultatele măsurătorilor efectuate, vom decide asupra celei mai bune alternative.

Cercetarea noastră propune o nouă metodă pentru selectarea componentelor, metodă bazată pe metrici şi analiza nuanţată.

5.3 Un algoritm nou bazat pe metrici şi clasificarea nuanţată pentru selectarea componentelor

5.3.1 Descrierea algoritmului

Această secţiune prezintă un nou algoritm pentru construirea unui sistem soft prin asamblarea de componente. În acest context, vom aborda problema selectării componentelor. Informal, problema noastră constă în selectarea unei submulţimi de componente care să satisfacă cerinţele sistemului. Dificultatea constă în faptul că fiecare componentă aparţine unei submulţimi de componente care au în comun anumite funcţionalităţi, iar scopul nostru este de a determina cea mai bună soluţie. Astfel, selectarea componentelor se va baza pe valorile metricilor propuse pentru cuantificarea atributelor de calitate stabilite precum şi pe partiţionarea componentelor folosind metoda de clasificare nuanţată.

Există două abordări alternative pentru algoritmul propus:

- o abordare care foloseşte doar o partiţie nuanţată, partiţie determinată pe baza cerinţelor iniţiale ale sistemului SR;
- a doua alternativă recalculează valorile metricilor pe baza cerinţelor reactualizate ale sistemului şi face o reclasificare a componentelor la fiecare pas al selecţiei.

Acest algoritm este descris în teză în Secţiunea 5.3.1.

5.3.2 Analiza comparativă cu alte abordări

Pentru a compara abordarea noastră cu alte abordări din literatura de specialitate, am propus un studiu de caz [SVP09] descris în teză în Secţiunea 5.3.2. Tabelul 1 prezintă soluţiile obţinute de toate abordările analizate. Analizând datele conţinute în acest tabel putem concluziona că soluţiile obţinute de algoritmul propus de noi sunt comparabile cu cele furnizate de celelalte abordări, având un avantaj major, şi anume: dimensiunea spaţiului de căutare este drastic redusă, partiţia rezultată sugerând componenta ce trebuie aleasă la fiecare pas; timpul de execuţie necesar pentru selecţia celei mai bune componente este redus din cauza limitării spaţiului de căutare;

Algoritm	Soluţia	*cost*	*reutilizabilitate*
Greedy	c_4, c_0, c_7, c_1	35	5
Branch and Bound	c_4, c_2, c_6, c_1	34	3
Algoritm Genetic (doar cost)	c_2, c_6, c_8	28	2
Algoritm Genetic (doar PSU şi RSU)	c_0, c_7, c_8	29	4
MFbCSwSPA	c_0, c_1, c_5, c_8	32	5
MFbCSwCPA	c_2, c_6, c_8	28	2

Tabela 1: Soluţiile obţinute de abordările analizate

5.4 Un cadru conceptual pentru evaluarea sistemelor bazate pe componente

În această secţiune propunem o abordare formală cu privire la evaluarea sistemelor bazate pe componente. Mai exact, scopul nostru este de a defini un cadru general, scalabil şi integrat pentru o evaluarea cantitativă atât a componentelor individuale cât şi a sistemului obţinut prin asamblarea lor. Cadrul propus este format din patru nivele de abstractizare: Un model pentru CBS, Obiectivele evaluării, Definiţiile formale ale metricilor, Analiza rezultatelor. Următoarele secţiuni descriu pe scurt aceste elemente.

5.4.1 Un model pentru sistemele bazate pe componente

Activitatea de măsurare trebuie să fie precedată de specificarea elementelor care vor fi evaluate, a proprietăţilor lor precum şi a relaţiilor care pot exista între aceste elemente. În felul acesta vom defini un model pentru sistemul analizat.

Definiţia 5.4.1 ([SVP10b]) *Tripletul* $(E, Prop(E), Rel(E))$ *defineşte un model pentru un sistem bazat pe componente, unde:*

- E *reprezintă mulţimea de entităţi ale sistemului;*
- $Prop(E)$ *reprezintă proprităţile elementelor din mulţimea* E*;*
- $Rel(E)$ *reprezintă relaţiile definite între entităţile din* E*.*

Elementele $E, Prop(E), Rel(E)$, referite în Definiţia 5.4.1 vor fi prezentate pe scurt în cele ce urmează.

Entităţi ale sistemului. Considerăm un depozit de componente $CR = \{c_1, c_2, ..., c_{noCR}\}$. O submulţime de componente $Comp(S) = \{c'_1, c'_2, ..., c'_{noComp}\}$, $Comp(S) \subseteq CR$ este selectată pentru a construi un sistem soft S care oferă serviciile $Serv(S) = \{s_1, s_2, ..., s_{noServ}\}$.

Fiecare componentă $c \in CR$ este specificată prin:

- mulţimea *interfeţelor oferite* $PI_c = \{pi_{c,1}, pi_{c,2}, ..., pi_{c,noPI_c}\}$;
- mulţimea *interfeţelor cerute* $RI_c = \{ri_{c,1}, ri_{c,2}, ..., ri_{c,noPI_c}\}$;
- dependenţele (contextul) între interfeţele cerute şi cele oferite.

Proprietăţi ale entităţilor sistemului. În această abordare au fost identificate trei tipuri de entităţi ale unui sistem bazat pe componente: componentă, interfaţă şi parametru. Pentru specificarea proprietăţilor acestor tipuri de entităţi am folosit modelul propus în teză în Secţiunea 5.4.1.2.

Relaţii între entităţile sistemului. Două tipuri de relaţii (de dependenţă şi de conexiune) au fost definite între entităţile sistemului. Aceste relaţii sunt descrise în teză în Secţiunea 5.4.1.3.

5.4.2 Obiectivele evaluării

Când construim un sistem soft bazat pe componente trebuiesc îndeplinite două tipuri de cerinţe : în primul rând, sistemul trebuie să ofere o mulţime de servicii/funcţionalităţi, iar în al doilea rând trebuie să îndeplinească anumite cerinţe non-funcţionale sau

atribute de calitate cum ar fi securitate, performanţa. Prin urmare, obiectivele principale în evaluarea sistemelor bazate pe componente sunt legate de aceste doua tipuri de cerinţe.

Verificarea dacă sistemul îndeplineşte cerinţele cu privire la funcţionalităţile sau serviciile ce trebuie să le ofere, nu ridică probleme. Astfel, cercetarea noastră propune o abordare cantitativă pentru stabilirea obiectivelor non-funcţionale, abordare bazată pe modelul Factor-Criteriu-Metrică [BR88]. Prin urmare, fiecare atribut de calitate este descompus în mai multe criterii, fiecare criteriu sugerând anumite metrici. În consecinţă, fiecare atribut de calitate va fi asociat cu una sau mai multe metrici care reflectă cel mai bine înţelesul său.

Este foarte uşor de observat că, calitatea componentelor individuale influenţează, direct sau indirect, calitatea sistemului final. Raţionamentul de compoziţie ne arată că proprietăţile sistemului sunt influenţate mai mult de interacţiunea dintre componentele sale decât de proprietăţile unei singure componente. O evaluare mai relevantă a componentelor poate fi obţinută în contextul sistemului în care sunt integrate [GA04b]. Aşadar, obiectivele evaluării se vor raporta la produsul ţintă: ansamblu de componente.

O altă problemă legată de atributele de calitate este aceea că ele se influenţează reciproc în mai multe moduri; de exemplu o creştere a unui atribut (uşurinţa de întreţinere) poate diminua al atribut (performanţa). Astfel, pentru fiecare dintre metricile selectate vom stabili o pondere (wf) cu rolul de a stabili prioritatea unui atribut de calitate în procesul de selecţie. Prin urmare, între doua componente care oferă servicii/funcţionalităţi similare, este posibil să selectam o componentă cu un grad mic de reutilizabilitate în schimbul unei componente cu un grad mare de reutilizabilitate dacă un alt atribut de calitate este mai important pentru client.

5.4.3 Definiţii formale ale metricilor

În Secţiunea 5.4.3 din teză sunt prezentate câteva metrici pentru dezvoltarea bazată pe componente. Formalizarea definiţiilor acestor metrici se bazează pe modelul propus în Secţiunea 5.4.1.

5.4.4 Analiza rezultatelor evaluării

Rezultatele evaluării sunt descrise formal după cum urmează:

- CR, mulţimea de entităţi evaluate;
- fieacare componentă $c_i \in CR$ se va identifica cu un vector $c_i = (c_{i_1}, c_{i_2}, ..., c_{i_{noM}})$ al valorilor metricilor $m_1, m_2, ..., m_{noM}$, $m_j \in Metrics(S)$, $1 \leq j \leq noM$ pentru componenta în discuţie.

O interpretare a acestor rezultate este necesară, în scopul de a o folosi ca dată de intrare în algoritmul de selectare a componentelor. Aşa cum am afirmat anterior cu privire la interpretarea rezultatelor, problema cea mai dificilă constă în stabilirea valorilor de prag pentru metrici. Pentru a depăşi această limitare, am utilizat metoda de analiză bazată pe clasificarea nuanţată. Acesta ne permite să plasăm un obiect în mai mulţi clusteri, având grade diferite de apartenenţă. Mai mult, metoda ne oferă posibilitatea de a lua în considerare, atunci când trebuie să decidem între două componente similare, elemente specifice ale sistemului analizat.

În cele ce urmează, vom introduce câteva definiţii pentru a arăta modul în care aplicăm analiza bazată pe clasificarea nuanţată în evaluarea propusă.

Definiţia 5.4.3 *Mulţimea* $U = \{U_1, U_2, ..., U_k\}$ *se numeşte* ***partiţie nuanţată*** *a mulţimii componentelor* $CR = \{c_1, c_2, ..., c_{noComp}\}$*, fiecare componentă* $c_i \in CR$ *identificându-se cu vectorul* $c_i = (c_{i_1}, c_{i_2}, ..., c_{i_{noM}})$ *al valorilor metricilor* $m_1, m_2, ..., m_{noM}$*,* $m_j \in Metrics(S)$*,* $1 \leq j \leq noM$*, ddacă:*

- $U_i = (u_{i1}, u_{i2}, ..., u_{i(noComp)})$*,* $1 \leq i \leq k$*;*
- $u_{ij} \in [0..1]$*,* $1 \leq i \leq k$*,* $1 \leq j \leq noComp$*,* u_{ij} *reprezintă gradul de apartenenţă al componentei* c_j *la clusterul* i*;*
- $\sum_{i=1}^{k} u_{ij} = 1$.

Algoritmul genetic de clasificare nuanţată [Bez81] folosit pentru obţinerea acestei partiţii, numit Fuzzy c-means are dezavantajul că numărul de clusteri este o dată de intrare. Din acest motiv, vom considera algoritmul Fuzzy Divisive Hierarchic Clustering (FDHC) [Dum88] pentru determinarea partiţiei optime. Partiţia obţinută în urma aplicării acestui algoritm este definită mai jos.

Definiţia 5.4.4 ([SVP10b]) *Mulţimea* $BTFP = \{N_1, N_2, ..., N_l\}$ *se numeşte arbore binar de partiţie nuanţată a mulţimii de componente* $CR = \{c_1, c_2, ..., c_{noComp}\}$*, fiecare componentă* $c_i \in CR$ *fiind identificată cu un vector* $c_i = (c_{i_1}, c_{i_2}, ..., c_{i_{noM}})$ *al valorilor metricilor* $m_1, m_2, ..., m_{noM}$*,* $m_j \in Metrics(S)$*,* $1 \leq j \leq noM$*, ddacă:*

- $\forall i \in \{1, 2, ..., l\}$*,* $N_i = (f_i, U_i)$*,* f_i*- tatăl clusterului* i*,* $f_i \in \{0...l\}$*;* $U_i = (u_{i1}, u_{i2}, ..., u_{i(noComp)})$*,* $u_{ij} \in [0..1]$*,* u_{ij} *reprezintă gradul de apartenenţă al componentei* c_j *la clusterul* i*;*
- $\exists i, i \in \{1, 2, ..., l\}$ *astfel încât* $f_i = 0 \wedge u_{ij=1}, \forall j \in \{1, 2, ..., noComp\}$*,* N_i *- nodul rădăcină;*
- $\forall i \in \{1...l\}$*,* $f_i \neq 0$*,* $\exists!\ j, k : f_i = k \wedge f_j = k \wedge U_i + U_j = U_k$*, unde* $U_i + U_j = (u_{i1} + u_{j1}, u_{i2} + u_{j1}, ..., u_{i(noComp)} + u_{j(noComp)})$.

Definiţia 5.4.5 ([SVP10b]) *Considerăm arborele binar de partiţie nuanţată* $BTFP = \{N_1, N_2, ..., N_l\}$*. Un nod* $N_i = (f_i, U_i) \in BTFP$ *se numeşte* ***nod terminal*** *sau* ***nod frunză*** *ddacă:*

$$\forall j \in \{1, 2, ..., l\}, j \neq i \text{ si } N_j = (f_j, U_j) \in BTFP \Rightarrow f_j \neq i.$$

Definiţia 5.4.6 ([SVP10b]) *Considerăm un arbore binar de partiţie nuanţată* $BTFP$*. O submulţime* $OFP = \{G_1, G_2, ..., G_k\} \subset BTFP$ *se numeşte* ***partiţie nuanţată optimă*** *ddacă:* G_j *este nod terminal din* $BTFP$*,* $\forall j \in \{1, 2, ..., k\}$.

Pornind de la partiţiile nuanţate descrise de definiţiile de mai sus, am construit un algoritm pentru problema selectării componentelor. Acest algoritm este descris în Secţiunea 5.3 din teză.

5.5 Concluzii

În acest capitol am propus o nouă abordare bazată pe metrici cu privire la evaluarea sistemelor bazate pe componente. Abordarea propusă are ca scop contracararea unor probleme întâlnite în activitatea de măsurare a softului cum ar fi:

- *definiţiile metricilor*; utilizând cadrul conceptual propus, metricile pot fi definite într-o manieră formală, definiţiile fiind neambigue, simple şi independente de limbaj.
- *interpretarea rezultatelor măsurătorilor*; metoda de analiză bazată pe clasificarea nuanţată este folosită pentru a contracara problema stabilirii valorilor de prag pentru metrici şi pentru selectarea componentelor.

Rezultatele originale ale acestui capitol au fost diseminate în articolele [SVP08, SVP09, SVP10a, SVP10b].

6 Evaluarea ansamblului de componente

În capitolul precedent am abordat problema selectării componentelor, selecţie bazată pe metrici. Soluţia obţinută are anumite limitări. De exemplu, nu ţine cont de interacţiunea dintre componente, deoarece acest aspect nu este cunoscut în totalitate decât după ce întregul sistem este construit. În general calitatea sistemului depinde de interacţiunea dintre componente şi ca urmare scopul nostru este de a evalua sistemul ca un întreg, facilitând astfel şi comparaţia cu diferite soluţii alternative.

În acest sens, capitolul curent propune o evaluare focalizată pe ansamblu de componente. Suntem interesaţi de cuplarea sistemului deoarece cuplarea vizează în principal interacţiunea dintre componente şi este strâns legată de atributele de calitate. Au fost selectate metrici şi au fost propuse altele noi pentru a cuantifica interacţiunea dintre componente şi pentru a studia influenţa valorilor metricilor asupra atributelor de calitate. Evaluarea propusă oferă suport pentru selecţia unei soluţii optime dintr-o mulţime de configuraţii ce reprezintă soluţii posibile.

6.1 Ansamblu de componente modelat ca un graf

Definiţia 6.1.1 ([SV07b]) *Un **ansamblu** este o relaţie binară notată prin* $DR = (C, D), D \subseteq C \times C$*, unde* C *este o mulţime de componente şi* D *este relaţia care conţine dependenţele dintre componente. Există o componentă* $c_0 \in C$ *cu un rol special: de a porni execuţia sistemului soft.*

Definiţia 6.1.2 ([SV07b]) *O **dependenţă** este o pereche* $d = (c_1, c_2) \in D$ *cu semnificaţia că pentru execuţia componentei* c_1 *sunt necesare serviciile oferite de* c_2 *(cu alte cuvinte,* c_1 *depinde de* c_2*).*

În această abordare, un sistem bazat pe componente (ansamblu de componente) este modelat ca un graf orientat (DR) în care nodurile sunt componentele (mulţimea C) iar arcele sunt dependenţele (mulţimea D) dintre componente. Astfel, fiecare ansamblu de componente este un graf orientat.

Utilizând graful orientat pentru un ansamblu este dificilă determinarea adâncimii şi "lăţimii" dependenţelor dintre componente. O mai bună vizualizare implică transformarea grafului orientat într-un arbore. Construcţia arborelui de dependenţă [SV07b] este descrisă în detaliu în teză ***D**ependency **T**ree **A**lgorithm* (**DTA**).

Pentru a obţine un arbore optimal care să conţină toate drumurile din arborele de dependenţă, un alt algoritm care să completeze arborele iniţial este aplicat. Algoritmul propus este denumit ***C**omplete **D**ependency **T**ree **A**lgorithm* (**CDTA**).

6.2 Metrici adaptate şi definite

În cele ce urmează sunt prezentate metricile existente în proiectarea orientată obiect [LK94, CK93] care au fost adaptate pentru ansamblul de componente.

Într-o proiectare orientată obiect, cuplarea reprezintă "interconexiunile dintre părţi" [CY91]. Declararea unui obiect într-o clasă externă creează o potenţială colaborare între cele două clase. Acest aspect este cuantificat prin metrica CBO (Coupling Between Objects) [CK93].

Considerăm un ansamblu de componente, $DR = (C, D)$, unde C este o mulţime de componente şi D este relaţia care conţine dependeţele dintre componente.

Definiţia 6.2.1 ([SV07b]) *A componentă* c_1 ***este cuplată*** *cu componenta* c_2 *dacă* $(c_1, c_2) \in D$.

Definiţia 6.2.2 ([SV07b]) Metrica **Coupling Between Components (CBC)** *a componentei* $c \in C$ *este numărul de componente cu care componenta c este cuplată.*

$$CBC(c) = card(\{d \in C | (c, d) \in D\}).$$

Vom studia cuplarea dintre componente pentru a evalua calitatea sistemului. Este bine cunoscut faptul că o cuplare excesivă are un rol negativ asupra multor atribute de calitate externe: *reutilizabilitatea, modularitatea, uşurinţa de înţelegere* şi *testabilitatea.*

Definiţia 6.2.3 ([SV07b]) Metrica **Depth Dependency Tree (DDT) a unei componente** c_0**.** *Considerăm o componentă* $c_n \in C$ *şi lanţul elementar* $c_0, c_1, ..., c_n$*, unde* c_0 *este nodul rădăcină. Valoarea metricii componentei* c_0 *este* $DDT(c_0) = n$*. Cu alte cuvinte, DDT măsoară lungimea lanţului de dependenţe de la o componentă dată la rădăcină.*

Definiţia 6.2.4 ([SV07b]) Metrica **Breadth Dependency Tree (BDT)** *reprezintă numărul de lanţuri de dependenţe de la rădăcină la toate frunzele arborelui.*

Aceste metrici sunt evaluate luând în considerare impactul lor asupra atributelor de calitate. Din acest punct de vedere, o valoare mare a metricii DDT face ca componenta să fie greu de *reutilizat* într-un alt context. În plus, *uşurinţa de înţelegere*, *întreţinerea* şi *testabilitatea* sunt de asemenea afectate. Aceasta deoarece, înţelegerea unei entităţi presupune înţelegerea recursivă a tuturor componentelor de care aceasta depinde. Mai mult, orice modificare a componentei c necesită modificări ale componentelor de care componenta c depinde. O valoare mare a metricii afectează *întreţinerea* şi alte criterii ca *uşurinţa de înţelegere*. Sistemul tinde să devină din ce în ce mai complex.

Secţiunea 6.1.4 conţine un exemplu, PDA - Personal Digital Assistant, pentru a discuta metricile adaptate şi cele introduse.

6.3 Selectarea unui ansamblu de componente pe baza metricilor

Atund când se construieşte un sistem soft dintr-o mulţime de componente, pot exista două abordări posibile:

1. prima abordare vizează construirea doar a soluţiei care îndeplineşte cerinţele sistemului, soluţie cât mai apropiată de cea optimă, adopând metode euristice;

2. cea de-a doua abordare vizează construirea tuturor soluţiilor posibile şi evaluarea fiecăreia pentru a selecta în final pe cea mai apropiată de scopul nostru.

Dacă în capitolul precedent ne-am ocupat de prima abordare, capitolul curent adoptă cea de-a doua soluţie. Metricile soft centrate pe evaluarea ansamblului de componente sunt folosite în procesul de selecţie a celei mai bune soluţii care răspunde cerinţelor sistemului.

6.3.1 Metricile propuse

Următoarele două metrici au fost propuse pentru a măsura cuplarea dintre componente.

Definiţia 6.3.1 ([SV07a]) **Component Coupling Grade.** Metrica *Component Coupling Grade (CCG) a unei componente X care depinde de o componentă Y, reprezintă numărul de servicii oferite de Y pe care X le utilizează. În cele ce urmează vom nota această valoare cu CCG(X,Y).*
Definiţia 6.3.2 ([SV07b]) **Component Coupling Total Grade.** Măsura *Component Coupling Total Grade (CCTG) a unei componente X care depinde de o mulţime de componente C_1,C_2,...,C_n, reprezintă numărul de servicii oferite de aceste componente şi pe care componente X le utilizează.*

$$CCTG = CCG(X, C_1) + CCG(X, C_2) + ... + CCG(X, C_n). \quad (1)$$

6.3.2 Influenţa valorilor metricilor asupra atributelor de calitate

Scopul nostru este de a defini metrici care sa fie relevante în măsurarea atributelor de calitate de care suntem interesaţi. Avem nevoie de aceste informaţii pentru a alege soluţia care reprezintă cel mai bine cerinţele sistemului.

Influenţa valorilor metricilor asupra atributelor de calitate (pe care le considerăm importante pentru evaluarea ansamblului) este prezentată în Tabelul 2. Următoarele notaţii sunt folosite: m pentru o valoare mică a metricii, M pentru o valoare mare a metricii, "+" pentru o influenţă pozitivă şi "−" pentru o influenţă negativă. De exemplu, o valoare mică a metricii IDC influenţează pozitiv reutilizabilitatea componentei.

	Reutilizabilitatea	Funcţionalitatea	Uşurinţa de înţelegere	Întreţinerea	Testabilitatea
PSU	m/+	m/-	m/+	m/+	m/+
RSU	m/+	-	m/+	m/+	m/+
CPSU	m/+	m/-	m/+	m/+	m/+
CRSU	m/+	-	m/+	m/+	m/+
IDC	m/+	m/-	m/+	m/+	m/+
IIDC	m/+	-	m/+	m/+	m/+
OIDC	m/+	m/-	m/+	m/+	m/+
AIDC	m/+	-	m/+	m/+	m/+
CCG	M/-	M/+	M/-	M/-	M/-
CCTG	M/-	M/+	M/-	M/-	M/-

Tabela 2: Influenţa valorilor metricilor asupra atributelor de calitate

O valoare prag este o limită (mare sau mică) pentru o anumită metrică. Toate valorile metricilor de mai sus iau valori între 0 şi 1, cu excepţia CCTG şi CCG. Valoarea pragului a fost fixată la 0.5.

Un exemplu care să discute metricile propuse este prezentat. În exemplul dat, nouă componente au fost determinate ca şi componente candidat. Alte două componente vor fi adăugate pentru a completa sistemul final: o componentă *Read* (R) şi o componentă $Write$ (W). Algoritmii utilizaţi [FMD06, VM06] determină mai multe soluţii. În cele ce urmează sunt prezentate doar două soluţii şi valorile metricilor pentru fiecare soluţie şi influenţa lor asupra atributelor de calitate.

Valorile metricilor pentru prima soluţie sunt în jurul valorii medii, pentru toate atributele de calitate. De exemplu, majoritatea componentelor au o foarte bună

funcţionalitate în sistem şi în acelaşi timp pot oferi noi funcţionalităţi pentru îmbunătăţiri viitoare prin adăugarea de noi servicii (influenţează atributul întreţinere). Referitor la metrica de cuplare, putem remarca că aceasta nu este foarte ridicată şi întreţinerea şi reutilizabilitatea nu sunt influenţate foarte tare. Valorile metricilor pentru ansamblu sugerează ca soluţia nu este considerată cea mai bună pentru fiecare atribut de calitate, ci o soluţie în medie cea mai bună pentru sistemul întreg. Valoarea metricii $AIDC$ este aproape de 1 dar trebuie considerată şi valoarea metricii $CCTG$ pentru a decide care soluţie reprezintă cel mai bine nevoile viitoare (dacă se doreşte o îmbunătăţire a sistemului şi să se adauge noi funcţionalităţi, sau dacă se doreşte o funcţionalitate bună în sistem).

Cea de-a doua soluţie conţine doar trei componente externe din mulţimea de componente candidat. Valorile metricilor care influenţează atributul funcţionalitate sunt aproape de 1, dezvăluind o bună funcţionalitate a fiecărei componente în sistem, dar celelalte valori ale metricilor influenţează negativ celelalte atribute de calitate. Valoarea prag aleasă 0.50 este depăşită de toate metricile calculate. În Tabelul 2 se poate observa că o valoare mare influenţează negativ aproape toate atributele de calitate discutate. Valoarea metricii $CCTG$ este relativ mare considerând că sunt puţine componente în soluţie. Valoarea $CCTG$ a celei de a noua componentă este mare influenţând înţelegerea şi testarea.

6.4 Concluzii

În acest capitol am abordat problema evaluării unui ansamblu de componente pe baza metricilor. Au fost adaptate metrici şi au fost definite metrici noi pentru a cuantifica interacţiunile dintre componente, studiind influenţa valorilor metricilor asupra atributelor de calitate. Abordarea propusă oferă suport în selectarea ansamblului de componente care reflectă cel mai bine cerinţele utilizatorului. Capitolul este bazat pe articolele diseminate în [SV07b, SV07a].

Concluzii şi direcţii viitoare de cercetare

Scopul principal al acestei teze, după cum a fost menţionat în partea introductivă, este acela de a investiga cum pot aplicate metricile soft în evaluarea sistemelor informatice, în particular în evaluarea sistemelor orientate obiect şi a sistemelor bazate pe componente. De asemenea, urmărim să investigăm metode şi tehnici care să ofere o interpretare relevantă a rezultatelor măsurătorilor.

În acest context, a fost introdusă *O metodologie de evaluare cantitativă pentru sistemele orientate obiect*. Metodologia propusă se bazează pe analiza statică a codului sursă şi este descrisă de un cadru conceptual compus din patru nivele de abstractizare după cum urmează: *i*) *Un model pentru proiectarea orientată obiect* – acest model defineşte formal domeniul evaluării, specificând elementele care sunt evaluate, proprietăţile lor şi relaţiile dintre ele; *ii*) *Definiţia formală a metricilor pentru proiectarea orientată obiect* – constă dintr-o bibliotecă de definiţii formale ale metricilor pentru proiectarea orientată obiect; *iii*) *Specificarea obiectivelor evaluării* – specifică într-o manieră formală obiectivele evaluării utilizând o abordare bazată pe metrici; *iv*) *Analiza rezultatelor măsurătorilor* – utilizează metoda de analiză bazată pe clasificarea nunaţată pentru interpretarea rezultatelor măsurătorilor.

O altă direcţie care a fost investigată este în domeniul sistemelor bazate pe componente. Selecţia unei componente (dintr-o mulţime de componente disponibile ce oferă functionalităţi similare) necesită evaluarea componentelor prin metode obiective. Rezultatele evaluării ajută dezvoltatorii în procesul de selecţie. Teza propune o abordare formală cu privire la evaluarea sistemelor bazate pe componente. Astfel, a fost definit un model general, scalabil şi integrat pentru o evaluare cantitativă, atât pentru componente individuale cât şi sistemul obţinut prin conectarea componentelor. Modelul propus asigură o terminologie standard pentru a defini formal metricile pentru dezvoltarea bazată pe componente. Metoda de analiză bazată pe clasificarea nuanţată a fost folosită ca şi metodă robustă pentru interpretarea rezultatelor măsurătorilor.

Noi metrici soft au fost introduse şi definite pentru evaluarea atributelor de calitate ale întregului sistem şi pentru alegerea celei mai bune soluţii dintr-un set de configuraţii disposibile. Au fost propuse unele atribute de calitate (reutilizabilitate, funcţionalitate, uşurinţa de întreţinere şi testabilitate) care sunt luate în considerare în analiza calităţii unui ansamblu de componente. Noi metrici soft (CCG - Component Coupling Grade, CCTG - Component Coupling Total Grade) sunt folosite pentru selectarea, din setul total de configuraţii obţinute, a soluţiei care satisface cel mai bine cerinţele sistemului.

Pentru fiecare abordare originală, am sugerat îmbunătăţiri posibile şi noi direcţii de cercetare în evaluarea sistemelor soft.

Bibliografie

[Abr93] F.B. Abreu. Metrics for Object Oriented Environment. In *Proceedings of the 3rd International Conference on Software Quality, Tahoe, Nevada, EUA, October 4th - 6th*, pages 67–75, 1993.

[Abr95] F.B. Abreu. The MOOD Metrics Set. In *9th European Conference on Object-Oriented Programming (ECOOP'95) Workshop Metrics*, 1995.

[AC99] C. Alves and J. Castro. Pore: Procurement-oriented requirements engineering method for the component based systems engineering development paradigm. *Int. Conf. Software Eng. CBSE Workshop*, 1999.

[AC01] C. Alves and J. Castro. Cre: A systematic method for cots component selection. *In Brazilian Symposium on Software Engineering, Rio De Janeiro*, 2001.

[AR94] F.B. Abreu and Carapuca Rogerio. Candidate Metrics for Object- Oriented Software within a Taxonomy Framework. In *Journal of systems software 26*, pages 359–368, 1994.

[BA04] M. A. S. Boxall and S. Araban. Interface Metrics for Reusability Analysis of Components. In *Proceeding of Australian Software Engineering Conference ASWEC'2004*, pages 40 – 51, 2004.

[Bar02] A.L. Baroni. *Formal Definition of Object-Oriented Design Metrics*. PhD thesis, Ecole des Mines de Nantes, 2002.

[BBA02] A. Baroni, S. Braz, and F. Abreu. Using OCL to Formalize Object-Oriented Design Metrics Definitions. In *ECOOP'02 Workshop on Quantitative Approaches in OO Software Engineering*, 2002.

[BBeA03a] A. Baroni, S. Braz, and F. Brito e Abreu. A formal library for aiding metrics extraction. *Proceedings of ECOOP Workshop on Object-Oriented Re-Engineering*, 2003.

[BBeA03b] A. Baroni, S. Braz, and F. Brito e Abreu. Using OCL to formalize object-oriented design metrics defnitions. *Proceedings of ECOOP Workshop on Quantative Approaches in Object-Oriented Software Engineering*, 2003.

[BBM96] V. Basili, L. Briand, and W. Melo. A validation of object-oriented design metrics as quality indicators. 20(10):751–761, 1996.

[BDE99] J. Bansiya, C. Davis, and L. Etzkorn. An entropy-based complexity measure for object-oriented designs. *Theory an Practice of Object Systems.*, 5(2):111–118, 1999.

[BDW99] L. Briand, J. Daly, and J. Wust. A Unified Framework for Coupling Measurement in Object-Oriented Systems. *IEEE Transactions on Software Engineering*, 25(1):91–121, 1999.

[Bez81] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[BHSS06] Paul Baker, Mark Harman, Kathleen Steinhofel, and Alexandros Skaliotis. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. In *ICSM '06: Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 176–185, Washington, DC, USA, 2006. IEEE Computer Society.

[BK95] J.M. Bieman and B.K. Kang. Cohesion and Reuse in an Object-Oriented System. *ACM Symposium on Software Reusability*, 1995.

[BME07] G. Booch, R. A. Maksimchuk, and Michael W. Engel. *Object-Oriented Analysis and Design with Applications.* Addison-Wesley, 2007.

[Boo94] G. Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin Cummings, Redwood City, 2 edition, 1994.

[BR88] V. Basili and D. Rombach. The TAME project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6), 1988.

[BTV06] M.F. Bertoa, J.M. Troya, and A. Vallecillo. Measuring the Usability of Software Components. Journal of Systems and Software. *IEEE Software*, 79(3):427–439, 2006.

[CK93] S. R. Chidamber and C. F. Kemerer. A Metrics suite for Object Oriented design. *IEEE Transactions on Software Engineering*, 20(6):476 – 493, 1993.

[CK94] S.R. Chidamber and C.F. Kemerer. A Metric Suite for Object- Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[CL02] I. Crnkovic and M. Larsson. *Building Reliable Component-Based Software Systems.* Artech House publisher, 2002.

[Crn03] I. Crnkovic. Component-based Software Engineering - New Challenges in Software Development. In *Proceeding of the 25th International Conference on Information Technology Interfaces*, pages 9 – 18, 2003.

[CS01] Philip T Cox and Baoming Song. A Formal Model for Component-Based Software. In *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)*, pages 304–311, 2001.

[CS03] A. Chatzigeorgiou and G. Stephanides. Entropy as a Measure of Object-Oriented Design Quality. In *1st Balkan Conference on Informatics (BCI'2003), Thessaloniki, Greece, November 21-23*, 2003.

[CSSW04] I. Crnkovic, H. Schmidt, J. A. Stafford, and K. Wallnau. Message from the Chairs. In *Proceedings of The 6^{th} ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction*, pages 1 – 7, 2004.

[CXS04] Alexander Chatzigeorgiou, Spiros Xanthos, and George Stephanides. Evaluating Object-Oriented Designs with Link Analysis. In *Proceedings of the 26th International Conference on Software Engineering*, pages 23–28, 2004.

[CY91] P. Coad and E. Yourdon. *Object-Oriented Design*. Prentice Hall, London, 2 edition, 1991.

[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.

[DeM82] T. DeMarco. *Controlling Software Projects; Management, Measurement and Estimation*. Yourdan Press, New Jersey, 1982.

[Des] Software Design Pattern. `http://en.wikipedia.org/wiki/Software_design_pattern`.

[Dum88] Dan Dumitrescu. *Hierarchical pattern classification*. Fuzzy Sets and Systems 28, 1988.

[DW98] D. F. D'Souza and A. C. Wills. *Objects, Components, and Frameworks with UML: The Catalysis Approach*. Addison-Wesley, Reading, MA, 1998.

[EGH02] L. H. Etzkorn, S. Gholston, and W. E. Hughes. A semantic entropy metric. *Journal of Software Maintenance: Research and Practice.*, 14(4):293–310, 2002.

[EWEBBF] Mohamed El-Wakil, Ali El-Bastawisi, Mokhtar Boshr, and Ali Fahmy. Software Metrics - A Taxonomy. Technical report.

[FBB+99] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[FBR04] Michael Roy Fox, David C. Brogan, and Jr. Paul F. Reynolds. Approximating component selection. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 429–434. Winter Simulation Conference, 2004.

[Fen94] N. Fenton. Software Measurement: A Necessary Scientific Base. *IEEE Transactions on Softw. Engineering, 20(3)*, 1994.

[Fen95] N.E. Fenton. *Software Metrics: A Rigorous Approach.* International Thomson Computer Press, London, UK, 1995.

[FMD06] A. Fanea, S. Motogna, and L. Diosan. Automata-based Component Composition Analysis. *Studia Universitas Babes-Bolyai, Seria Informatica*, LI(1):13–20, 2006.

[FP97] N. Fenton and S.L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* International Thomson Computer Press, London, UK, second edition, 1997.

[FP02] M. Frentiu and H.F. Pop. A study of dependence of software attributes using data analisys techniques. *Studia Universitas Babes-Bolyai, Seria Informatica*, L(2):53–66, 2002.

[GA04a] Miguel Goulão and O Brito E Abreu. Formalizing Metrics for COTS. In *Proceddings of the ICSE Workshop on Models and Processes for the Evaluation of COTS Components*, pages 37–40, 2004.

[GA04b] Miguel Goulão and O Brito E Abreu. Software Components Evaluation: an Overview. In *In Proceedings of the 5th Conferencia da APSI*, 2004.

[GA05a] M. Goulao and F.B. Abreu. Formal Definition of Metrics upon the CORBA Component Model. In *First International Conference on the Quality of Software Architectures*, 2005.

[GA05b] M. A. Goulão and F. B. Abreu. Composition Assessment Metrics for CBSE. In *Proceedings of The 31st Euromicro Conference, Component-Based Software Engineering Track*, pages 96 – 103, 2005.

[GG03a] N. S. Gill and P. S. Grover. Component-based measurement: few useful guidelines. *SIGSOFT Softw. Eng. Notes*, 28(6):1–4, 2003.

[GG03b] N. S. Gill and P. S. Grover. Reusability Issues in Component-based Development. *SIGSOFT Softw. Eng. Notes*, 28(4):1–4, 2003.

[GG04] N. S. Gill and P. S. Grover. Few important considerations for deriving interface complexity metric for component-based systems. *SIGSOFT Softw. Eng. Notes*, 29(2):1–4, 2004.

[GG07] L. Gesellensetter and S. Glesner. Only the Best Can Make It: Optimal Component Selection . *Electronic Notes in Theoretical Computer Science*, 176(2):105–124, 2007.

[GHJV94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[Hai] System.Reflection-based ILReader. `http://blogs.msdn.com/b/haibo_luo/archive/2006/11/06/system-reflection-based-ilreader.aspx`.

[HDM03] A. v. d. Hoek, E. Dincel, and N. Medvidovic. Using Service Utilization Metrics to Assess and Improve Product Line Architectures . In *In Proceedings of the 9th IEEE International Software Metrics Symposium Metrics*, pages 0 – 0, 2003.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[HS96] B. Henderson-Sellers. Object-Oriented Metrics-Measures of Complexity. *Prentice Hall, Sydney*, 1996.

[JD98] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[JF88] R.E. Johnson and B. Foote. Designing reusable classes. *Journal of Object-Oriented Programming*, 1(2):22–35, 1988.

[JMF99] A. Jain, M. N. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Kel89] W. T. Kelvin. *Popular Lectures and Addresses*. 1889.

[Kon95] J. Kontio. OTSO: A Systematic Process for Reusable Software Component Selection. Technical report, Technical report, University of Maryland, 1995.

[KSW95] K. Kim, Y. Shin, and C. Wu. Complexity Measures for Object-Oriented Program Based on the Entropy. In *In Proceedings of the Second Asia Pacific Software Engineering Conference*, 1995.

[Lak96] J. Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, 1996.

[LD02] Michele Lanza and Stéphane Ducasse. Beyond Language Independent Object-Oriented Metrics: Model Independent Metrics. In F. Abreu, Mario Piattini, Geert Poels, and Houari A. Sahraoui, editors, *Proceedings of the 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, pages 77–84, 2002.

[LH89] K.J. Lieberherr and I.M. Holland. Assuring good style for object-oriented programs. *IEEE Software*, 6:38–48, 1989.

[LH93a] W. Li and S. Henry. Maintenance Metrics for the Object Oriented Paradigm. *IEEE Proc. First International Software Metrics Symp*, pages 52–60, 1993.

[LH93b] W. Li and S. Henry. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 23(2):111–122, 1993.

[Lis87] Barbara Liskov. Keynote address - data abstraction and hierarchy. In *OOPSLA '87 Addendum to the proceedings on Object-oriented programming systems, languages and applications (Addendum), NY, USA*, 1987.

[LK94] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics*. Prentice-Hall Object-Oriented Series, Englewood Cliffs, NY, 1994.

[LM06] M. Lanza and R. Marinescu. *Object-Oriented Metrics in Practice*. Springer Verlag, 2006.

[log] Open Source Project: log4net. `http://logging.apache.org/log4net`.

[Lor93] M. Lorenz. *Object-Oriented Software Development: A Practical Guide*. Prentice-Hall, NJ, 1993.

[LR06] M. Lippert and S. Roock. *Refactoring in Large Software Projects*. John Wiley & Sons, 2006.

[LTGP02] A. Lozano-Tello and A. Gomez-Perez. ABAREMO: how to choose the appropriate software component using the analytic hierarchy process. In *The 14th international conference on Software engineering and knowledge engineering*, pages 781–788, ACM , New York, 2002.

[Mar] Robert Martin. Design Principles and Patterns. `http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf`.

[Mar97] R. Marinescu. The Use of Software Metrics in the Design of Object-Oriented Systems. Technical report, Politehnica University Timisoara, 1997.

[Mar02] R. Marinescu. *Measurement and Quality in Object Oriented Design*. PhD thesis, Faculty of Automatics and Computer Science, University of Timisoara, 2002.

[Mar09] Cristina Marinescu. *Towards Understanding and Quality Assessment of Enterprise Software Systems*. PhD thesis, Faculty of Automatics and Computer Science, University of Timisoara, 2009.

[MBA04] A. Marcus, M. Boxall, and S. Araban. Interface Metrics for Reusability Analysis of Components. In *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)*, 2004.

[McC76] T.J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering, 2(4)*, pages 308–320, 1976.

[Mey88] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, Englewood Cliffs, 1988.

[Mic] Microsoft Corporation. Definition of the term component. `http://www.msdn.microsoft.com/repository/OIM/resdkdefinitionofthetermcomponent.asp`.

[Mih03] Petru Florin Mihancea. Optimizarea detectiei automate a carentelor de proiectare în sistemele software orientate pe obiecte. Technical report, Facultatea de Automatica si Calculatoare, Universitatea Tehnica Timisoara, 2003.

[ML02] T. Mens and M. Lanza. A graph-based metamodel for object-oriented software metrics. *Electronic Notes in Theoretical Computer Science*, 72:57–68, 2002.

[MP06] Jacqueline McQuillan and James Power. Towards the re-usability of software metrics definitions at the meta level. In *Proceedings of the ECOOP'2006 Doctoral Symposium*, 2006.

[MP08] Jacqueline A. McQuillan and James F. Power. A Metamodel for the Measurement of Object-Oriented Systems: An Analysis using Alloy. In *IEEE International Conference on Software Testing Verification and Validation*, pages 288–297, 2008.

[MSL06] S. Mazeiar, Li. Shimin, and T. Ladan. A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws. In *Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC06)*, 2006.

[NH04] V. L. Narasimhan and B. Hendradjaya. A New Suite of Metrics for the Integration of Software Components. In *The First International Workshop on Object Systems and Software Architectures WOSSA 2004*, 2004.

[OC08] Mark O'Keeffe and Mel Ó Cinnéide. Search-based refactoring: an empirical study. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(5):345–364, 2008.

[OCBZ09] Steffen Olbrich, Daniela S. Cruzes, Victor Basili, and Nico Zazworka. The Evolution and Impact of Code Smells: A Case Study of Two Open Source Systems. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, Orlando, Florida*, 2009.

[P.F05] P.F. Mihancea and R. Marinescu. Towards the optimization of automatic detection of design flaws in object-oriented software systems. In *In Proc. of the 9th European Conf. on Software Maintenance and Reengineering*, pages 92–101, 2005.

[Pfl98] S.L. Pfleeger. *Software Engineering – Theory and Practice*. Prentice Hall, 1998.

[Rei01] R. Reißing. Towards a model for object-oriented design measurement. *Proceedings of ECOOP Workshop on Quantative Approaches in Object-Oriented Software Engineering*, 2001.

[RH00] L. Rosenberg and L. Hyatt. Software Quality Metrics for Object-Oriented System Environments. Technical report, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2000.

[Rie96] A.J. Riel. *Object-Oriented Design Heuristics*. Addison-Wesley, 1996.

[RL92] C. Rajaraman and M.R. Lyu. Some Coupling Measures for C++ Programs. *Proceedings of TOOLS USA-92, Prentice-Hall, Englewood Cliffs, NJ*, 1992.

[Ros98] Linda H. Rosenberg. Applying and Interpreting Object Oriented Metrics. In *In Software Technology Conference (April 1998)*, 1998.

[SC06] C. Serban and C. Cretu. Impact on Design Quality of Refactorings on Code via Metrics. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 39–44, 2006.

[Sch03] Jean-Guy Schneider. *Component Scripts and Glue: A Conceptual framework for software composition*. PhD thesis, Institute für Informatik (IAM), Universität Bern, Berne, Switzerland, 2003.

[Ser06] C. Serban. Coupling measurement for compiled .Net code. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 21–26, 2006.

[Ser09a] C. Serban. A Formal Approach for OOD Metrics Definition. *First International Conference on Modelling and Development of Intelligent Systems, Sibiu, Romania*, pages 262–269, 2009.

[Ser09b] C. Serban. High Coupling Detection Using Fuzzy Clustering Analysis. *Knowledge Engineering: Principles and Techniques (Post-proceedings of KEPT 2009), International Conference, Babes-Bolyai University, Presa Universitara Clujeana*, pages 258 – 264, 2009.

[Ser09c] C. Serban. High Coupling Detection Using Fuzzy Clustering Analysis. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceeding of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 223 – 226, 2009.

[Ser10] C. Serban. A conceptual framework for object-oriented design assessment. In *UKSim 4th European Modelling Symposium on Mathematical Modelling and Computer Simulation, Pisa, 17 - 19 November*, pages 90–95, 2010.

[Ser11] C. Serban. God Class Design Flaw Detection In Object Oriented Design. A Case–Study. *Studia Universitas Babes-Bolyai, Seria Informatica*, LVI(4):33–38, 2011.

[Sha09] Arun Sharma. *Design and Analysis of Metrics for Component-Based Software Systems*. PhD thesis, School of Mathematics and Computer Applications, Thapar University, India, 2009.

[SKG07] Arun Sharma, Rajesh Kumar, and P. S. Grover. A Critical Survey of Reusability Aspects for Component-Based Systems. *World Academy of Science, Engineering and Technology*, 2007.

[SM05] C. Serban and A. Mihis. Software quality assurance. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 207–212, 2005.

[SP08] C. Serban and H.F. Pop. Software Quality Assessment Using a Fuzzy Clustering Approach. *Studia Universitas Babes-Bolyai, Seria Informatica*, LIII(2):27–38, 2008.

[Spa00] M. Sparling. Lessons Learned Through Six Years of Component–Based Development. 43(10):47–53, 2000.

[SRM+10] Amjan Shaik, R. K. Reddy, B. Manda, C. Prakashini, and K. Deepthi. An Empirical Validation of Object Oriented Design Metrics in Object Oriented Systems. *Journal of Emerging Trends in Engineering and Applied Sciences*, 2(1):216–224, 2010.

[SV07a] C. Serban and A. Vescan. Metrics-based selection of a component assembly. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceeding of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 324 – 331, 2007.

[SV07b] C. Serban and A. Vescan. Metrics for Component-Based System Development. *Creative Mathematics and Informatics*, pages 143 – 150, 2007.

[SVP08] C. Serban, A. Vescan, and H.F. Pop. Component Selection based on Fuzzy Clustering Analysis. *Creative Mathematics and Informatics*, 17(3):505 – 510, 2008.

[SVP09] C. Serban, A. Vescan, and H.F. Pop. A new component selection algorithm based on metrics and fuzzy clustering analysis. *In Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, pages 621–628, 2009.

[SVP10a] C. Serban, A. Vescan, and H. F. Pop. A conceptual framework for component-based system metrics definition. In *9th RoEduNet International Conference, June 2010, Sibiu, Romania*, pages 73–78, 2010.

[SVP10b] C. Serban, A. Vescan, and H. F. Pop. A Formal Model for Component-Based System Assessment. In *Second International Conference on Computational Intelligence, Modelling and Simulation, Bali, 28 - 30 September 2010*, pages 261–266, 2010.

[SW49] C.E. Shannon and W Weaver. *The Mathematical Theory of Communication*. Urbana, IL, University:University of Illinois Press, 1949.

[Szy98] C. Szyperski. *Component Software, Beyond Object-Oriented Programming*. ACM Press, Addison-Wesley, 1998.

[TS92] D.P. Tegarden and S.D. Sheetz. Object-oriented system complexity: an integrated model of structure and perceptions. *In OOPSLA92 Workshop on Metrics for Object-Oriented Software Development (Washington DC)*, 1992.

[TSM92] D. Tegarden, S. Sheetz, and D. Monarchi. Effectiveness of Traditional Software Metrics for Object-Oriented Systems. In *25th Hawaii International Confernce on System Sciences*, pages 359–368, 1992.

[Ves08a] A. Vescan. An evolutionary multiobjective approach for the Component Selection Problem. In *Proc. of the First IEEE International Conference on the Applications of Digital Information and Web Technologies, 4 - 6 August, Ostrava, Czech Republic*, pages 252–257, 2008.

[Ves08b] A. Vescan. *Construction Approaches for Component-based Systems*. PhD thesis, Department of Computer Science, Babes-Bolyai University, 2008.

[Ves09] A. Vescan. A Metrics-based Evolutionary Approach for the Component Selection Problem. *in Proceedings of the 11th International Conference on Computer Modelling and Simulation*, 2009.

[VM06] A. Vescan and S. Motogna. Syntactic automata-based component composition. In *The 32nd EUROMICRO Software Engineering and Advanced Applications (SEAA), Proceeding of the Work in Progress session*, pages 13–14, 2006.

[VP08] A. Vescan and H. F. Pop. The Component Selection Problem as a Constraint Optimization Problem. *in Software Engineering Techniques in Progress*, pages 203–211, 2008.

[WBD98] J. Wust, L. Briand, and J. Daly. A Unified Framework for Cohesion Measurement in Object-Oriented Systems, journal = Empirical Software Engineering: An International Journal. 3(2):65–117, 1998.

[Wet04] Richard Wettel. Automated Detection of Code Duplication Clusters. Technical report, Faculty of Automatics and Computer Science, Politehnica University of Timisoara, 2004.

[WM96] Arthur H. Watson and Thomas J. McCabe. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. In *National Institute of Standards and Technology NIST Special Publication*, pages 500–235, 1996.

[WYF03] H. Washizaki, H. Yamamoto, and Y. Fukazawa. A Metrics Suite for Measuring Reusability of Software Components. In *In Proceedings of 9th IEEE International Software Metrics Symposium METRICS 2003*, pages 211 – 223, 2003.

[XHC+00] T. Xie, H. Huang, X. Chen, H. Mei, and F. Yang. *Object Oriented Software Quality Evaluation Technology*. Department of Computer Science and Technology, Peking University, 2000.

[Zad65] L.A. Zadeh. *Fuzzy sets*. Information and Control 8, 1965.