**CONTRACTOR** **nr.contract: 477**
**Universitatea Babeş-Bolyai**

Programul: **IDEI**
Tipul proiectului: **Proiecte de cercetare exploratorie**
Cod proiect: **ID_2286**

Sinteza lucrării

# CERCETĂRI ÎN DIRECŢIA OPTIMIZĂRII ADAPTIVE A SISTEMELOR INFORMATICE FOLOSIND TEHNICI DE ÎNVĂŢARE AUTOMATĂ ŞI SISTEME MULTIAGENT

Etapa unică an 2010

## A. Sinteza cercetării. Rezultate obţinute

În cele ce urmează vom prezenta o sinteză a rezultatelor originale obţinute în urma cercetărilor efectuate în cadrul proiectului în scopul îndeplinirii obiectivelor ştiinţifice propuse în planul de realizare a proiectului pe anul 2010. Vom indica, pentru fiecare obiectiv prevăzut în planul de realizare pe anul 2010, modul în care au fost îndeplinite activităţile aferente. De asemenea, vom menţiona rezultatele originale obţinute în vederea realizării obiectivelor.

Menţionez faptul că obiectivele planificate pe anul 2010, cât şi activităţile aferente acestora au fost realizate în totalitate, şi desfăşurate conform cu planul de realizare al proiectului. De asemenea, criteriul minim de performantă prevăzut (**2** articole ISI şi **3** articole BDI) a fost îndeplinit, aşa cum se poate vedea în secţiunea legată de modul de diseminare al rezultatelor.

### 1. Dezvoltarea sistemului multiagent inteligent DSSEM de suport decizional pentru asistarea dezvoltatorilor în faza de întreţinere şi evoluţie a sistemelor informatice

Sistemul DSSEM este un sistem multiagent inteligent de suport decizional pentru asistarea dezvoltatorilor în faza de întreţinere şi evoluţie a sistemelor informatice. Domeniile în care sistemul oferă suport, alocate câte unui tip de agent inteligent, se referă la optimizarea structurală şi comportamentală a acestora: identificarea şabloanelor de proiectare, introducerea de şabloane de proiectare, identificarea funcţionalităţilor transversale, localizarea conceptelor, refactorizare şi adaptarea comportamentală a sistemelor.

Necesitatea modificărilor structurale şi comportamentale a unui sistem este determinată de adăugarea de noi cerinţe funcţionale. Efectuarea acestor modificări presupune realizarea a doi paşi esenţiali în ingineria soft: *înţelegerea programelor* şi *reingineria soft*. Activităţile din domeniul înţelegerii programelor care se intenţionează a fi automatizate sunt identificarea şabloanelor de proiectare, localizarea conceptelor şi **aspect mining**. Corespunzător reingineriei sistemelor informatice studiem probleme legate de refactorizare şi introducerea de şabloane de proiectare. Adaptarea comportamentului sistemelor informatice presupune capacitatea de învăţare, fapt care determină necesitatea folosirii tehnicilor de învăţare. Adaptarea comportamentului unui sistem cuprinde aspecte precum optimizarea comunicării cu alte sisteme sau adaptarea structurilor de date la fluxul de date, etc.

#### 1.1 Modelarea conceptuală folosind tehnologia bazată pe agenţi a unui sistem care să permită optimizarea adaptivă (structurală şi comportamentală) a sistemelor informatice

La nivelul sistemului (distribuit) au fost identificate următoarele categorii de agenţi: *agenţi umani* reprezentaţi de dezvoltatorii de sisteme soft; şi *agenţi personali* - agenţi de interfaţă care colectează informaţii legate de codul sursă al unei aplicaţii dezvoltate într-un limbaj de programare

(noi am ales limbajul Java, fără a reduce însă generalitatea abordării). Arhitectura sistemului propus este o arhitectură distribuită care constă în mai multe noduri locale care reprezintă locaţiile în care dezvoltatorii de soft sunt asistaţi de către sistemul DSSEM şi un nod central care supervizează activităţile desfăşurate în nodurile locale.

La nivelul fiecărui nod local, agentul uman foloseşte mediul de dezvoltare preferat pentru dezvoltarea unei aplicaţii soft notate prin SA. La acest nivel un agent personal (PA) este disponibil şi este activat de fiecare dată când agentul uman are nevoie de asistenţă în sarcini precum identificarea şabloanelor de programare, refactorizare, localizare de concepte, identificare de funcionalităţi transversale, introducere de noi şabloane de proiectare sau adaptare comportamentală a sistemului soft. Agentul PA colectează informaţii despre codul sursă prin interacţiune cu mediul de dezvoltare în care agentul uman programează. La nivelul central sunt disponibile şase categorii de agenţi decizionali, corespunzători fiecăreia din activităţile de întreţinere şi evoluare a sistemelor soft menţionate anterior, iar din fiecare categorie este posibil să existe instanţe multiple de agenţi. Aceşti agenţi sunt activaţi de către agentul personal în funcţie de activitatea în care agentul uman are nevoie de asistenţă. Odată activată, o instanţă a corespunzătoare a agentului decizional (DSA) migrează la nodul local şi interacţionează cu agentul personal. După ce agentul DSA îşi finalizează activitatea, acesta trimite rezultatele agentului PA, care le transmite mai departe programatorului, iar agentul DSA migrează înapoi la nodul central. Agenţii DSA sunt agenţi inteligenţi care folosesc tehnici de instruire automată în vederea asistării programatorilor în sarcinile lor. Aceste tehnici inteligente au fost deja şi vor fi în continuarea dezvoltate în cadrul proiectului. Pentru modelarea sistemului multiagent s-a folosit instrumentul INGENIAS IDK [12] care permite dezvoltarea de specificaţii pentru sisteme multiagent. Specificarea unui sistem multiagent constă în realizarea unui set de diagrame care surprind viziuni diferite asupra sistemului multiagent. Specificaţiile realizate sunt salvate în format XML astfel încât să poată fi accesate şi de alte instrumente.

### 1.2 Definirea arhitecturii sistemului DSSEM astfel încât să permită dezvoltarea sa incrementală

Un prototip al sistemului multiagent DSSEM a fost implementat folosind platforma JADE (*Java Agent DEvelopment Framework*), un mediu implementat în Java care permite dezvoltarea sistemelor multiagent. Acesta are implementată funcţionalitatea de identificare a refactorizărilor prin intermediul agentului RDSA (Refactoring Decision Support Agent). RDSA este o instanţiere a agentului DSA în domeniul identificării refactorizărilor şi foloseşte o abordare bazată pe clustering în acest scop. Alegerea platformei JADE s-a decis în urma unei analize a caracteristicilor şi avantajelor oferite de acest mediu de dezvoltare aliniat standardelor FIPA [11] şi care dispune de un pachet de dezvoltare a agenţilor în limbajul de programare Java. Limbajul de programare Jade aderă la specificaţiile FIPA oferind astfel o largă interoperabilitate cu alte platforme care respectă specificaţiile FIPA. O platformă Jade este constituită din unul sau mai multe containere care pot fi distribuite în cadrul unei reţele. Containerele sunt entităţi în cadrul cărora există agenţii şi care furnizează infrastructura necesară pentru găzduirea şi execuţia agenţilor.

### 1.3 Adăugarea funcţionalităţilor de identificare a şabloanelor de proiectare, localizarea conceptelor şi identificarea aspectelor în sistemul DSSEM

După cum am arătat anterior, a fost implementat în JADE un prototip al sistemului DSSEM, în care au fost adăugate funcţionalităţile de identificare a şabloanelor de proiectare, localizar a conceptelor şi identificare a aspectelor. Aceste funcţionalităţi implementează algoritmii de clustering care au fost dezvoltaţi în cadrul proiectului în vederea automatizării activităţilor menţionate anterior. Arhitectura sistemului DSSEM este cea prezentată în secţiunile anterioare.

## 2. Elaborarea unor modele teoretice pentru probleme legate de reingineria soft: restructurare şi introducerea şabloanelor de proiectare în sisteme informatice existente

### 2.1 Elaborarea unui model matematic pentru problema introducerii şabloanelor de proiectare în sisteme informatice existente

Fie $S = \{C_1, C_2, ..., C_l\}$ un sistem informatic orientat obiect, unde $C_i, 1 \leq i \leq l$ este o clasă a sistemului.

Fără a restrânge generalitatea abordării, vom considera că vrem să identificăm în sistemul $S$ locaţiile unde ar trebui introdus un anumit şablon de proiectare $p$. Pentru început, ne-am orientat asupra şabloaneor de proiectare structurale.

Un *şablon de proiectare* $p$ din sistemul informatic $S$ poate fi privit ca fiind o pereche, $p = (\mathcal{C}_p, \mathcal{R}_p)$, unde

- $\mathcal{C}_p = \{C_1^p, C_2^p, \ldots, C_{nc_p}^p\}$ este o submulţime a mulţimii $S$, $\mathcal{C}_p \subset S$, şi reprezintă mulţimea claselor care fac parte din şablonul de proiectare $p$. $nc_p$ reprezintă numărul de clase din şablon.
- $\mathcal{R}_p = \{r_1^p, r_2^p, \ldots, r_{nr_p}^p\}$ este mulţimea restricţiilor (relaţiilor) existente între clasele din $\mathcal{C}_p$, restricţii care caracterizează şablonul de proiectare $p$. Ca urmare, fiecare restricţie $r_i^p, \forall 1 \leq i \leq nr_p$ din mulţimea $\mathcal{R}_p$ este o relaţie definită pe o submulţime de clase din $\mathcal{C}_p$, iar $nr_p$ reprezintă numărul de restricţii care caracterizează şablonul de proiectare $p$.

Menţionăm faptul că toate restricţiile din mulţimea $\mathcal{R}_p$ pot fi exprimate sub forma unor restricţii binare (sunt două clase participante la restricţie).

Problema identificării locaţiilor unde ar trebui introdus şablonul $p$ se poate reduce la problema identificării unor submulţimi de clase din $S$ al căror comportament este similar cu cel indicat de şablonul $p$. Pentru a ilustra acest grad de similaritate, am introdus o măsură $Sim$ care indică similaritatea între o mulţime de clase $\{Cl_1, ..., Cl_k\} \subset S$ şi şablonul $p = (\mathcal{C}_p, \mathcal{R}_p)$.

Ca urmare, problema studiată am redus-o la a determina submulţimi ale lui $S$ a căror similaritate $Sim$ în raport cu $p$ este mai mare decât un prag $Prag$ dat. Pentru rezolvarea acestei probleme, care poate fi văzută de fapt ca şi o problemă de satisfacere a restricţiilor, am introdus un algoritm de căutare sistematică de tip BackJumping.

Privită din perspectiva problemei de a introduce şabloane de proiectare în sisteme informatice existente, am studiat în [8, 10] problema automatizării procesului de transformare a programelor procedurale în programe orientate obiect. Transformarea programelor procedurale în arhitecturi orientate obiect este un proces deosebit de important pentru a asigura refolosirea programelor procedurale. Am dezvoltat un algoritm de clustering ierarhic aglomerativ care poate fi folosit pentru a asista dezvoltatorii soft în procesul de transformare a codului procedural într-unul orientat obiect.

### 2.2 Dezvoltarea unui model matematic pentru problema identificării refactorizărilor

Fie $S = \{s_1, s_2, ..., s_n\}$ un sistem informatic orientat obiect, unde $s_i, 1 \leq i \leq n$ poate fi o clasă, o metodă dintr-o clasă sau un atribut al unei clase. Ne vom referi la un element al sistemului informatic $S$ ca la o *entitate*.

Să considerăm următoarele:

- $Class(S) = \{C_1, C_2, \ldots, C_l\}$, $Class(S) \subset S$, este mulţimea claselor din sistemul informatic $S$.
- Fiecare clasă $C_i$ $(1 \leq i \leq l)$ este o mulţime de metode şi atribute, altfel spus, $C_i = \{m_{i1}, m_{i2}, \ldots, m_{ip_i}, a_{i1}, a_{i2}, \ldots, a_{ir_i}\}$, $1 \leq p_i \leq n$, $1 \leq r_i \leq n$, unde $m_{ij}$ $(\forall j,\ 1 \leq j \leq p_i)$ sunt metodele şi $a_{ik}$ $(\forall k,\ 1 \leq k \leq r_i)$ sunt atributele din clasa $C_i$.
- $Meth(S) = \bigcup_{i=1}^{l} \bigcup_{j=1}^{p_i} m_{ij}$, $Meth(S) \subset S$, este mulţimea metodelor din toate clasele sistemului informatic $S$.
- $Attr(S) = \bigcup_{i=1}^{l} \bigcup_{j=1}^{r_i} a_{ij}$, $Attr(S) \subset S$, este mulţimea atributelor din toate clasele sistemului informatic $S$.

Pe baza notaţiilor anterioare, sistemul informatic $S$ poate fi definit ca în Ecuaţia (1):

$$S = Class(S) \bigcup Meth(S) \bigcup Attr(S). \quad (1)$$

În această secţiune vom prezenta modelul teoretic pe care l-am dezvoltat pentru problema identificării refactorizărilor necesare în scopul îmbunătăţirii structurii de clase a unui sistem informatic orientat obiect. In accepţiunea noastră, problema de identifcare a refactorizărilor se reduce la regruparea entităţilor din sistem, fiind reprezentată sub forma unei partiţii a lui $S$.

**Definiţia 1** ***Partiţie a sistemului informatic*** $S$.
*Mulţimea* $\mathcal{K} = \{K_1, K_2, ..., K_v\}$ *se numeşte o* ***partiţie*** *a sistemului informatic* $S = \{s_1, s_2, \dots, s_n\}$ *ddacă*

- $1 \leq v \leq n$;
- $K_i \subseteq S, K_i \neq \emptyset, \forall i,\ 1 \leq i \leq v$;
- $S = \bigcup_{i=1}^{v} K_i$ *and* $K_i \cap K_j = \emptyset,\ \forall i, j, 1 \leq i, j \leq v, i \neq j$.

Un element $K_i (1 \leq i \leq v)$ din partiţia $\mathcal{K}$ îl vom numi cluster şi el va corespunde unei clase a sistemului informatic. În vederea restructurării sistemului informatic, ne propunem regruparea entităţilor similare din $S$ cu scopul de a obţine grupuri (clusteri) puternic coezive. În acest scop vom adapta măsura generică de coeziune, măsură care este legată de teoria similarităţii şi disimilarităţii. În opinia noastră, această măsură de coeziune este cea mai potrivită scopului nostru. Vom considera gradul de disimilaritate (din punct de vedere al coeziunii) între două clase ale sistemului informatic $S$. Ca urmare, vom considera *distanţa* $d(s_i, s_j)$ dintre două entităţi $s_i$ şi $s_j$ ale sistemului $S$ exprimată ca în Ecuaţia (2).

$$d(s_i, s_j) = \begin{cases} 1 - \frac{|p(s_i) \cap p(s_j)|}{|p(s_i) \cup p(s_j)|} & if \quad p(s_i) \cap p(s_j) \neq \emptyset \\ \infty & otherwise \end{cases}, \quad (2)$$

unde $p(e)$ defineşte o mulţime a proprietăţilor relevante ale entităţii $e$ şi este exprimată astfel:

- Dacă $e \in Attr(S)$ ($e$ este un atribut) atunci $p(e)$ constă în: atributul însuşi, clasa în care atributul a fost definit, precum şi toate metodele din $Meth(S)$ din care se accesează atributul.
- Dacă $e \in Meth(S)$ ($e$ este o metodă) atunci $p(e)$ constă în: metoda însăşi, clasa în care metoda a fost definită, precum şi toate atributele din $Attr(S)$ accesate de metodă.
- Dacă $e \in Class(S)$ ($e$ este o clasă) atunci $p(e)$ constă în: clasa însăşi, toate atributele şi metodele definite în clasă, toate interfeţele implementate de $C$ şi clasa de bază a clasei $C$.

Am ales distanţa (disimilaritatea) dinte două entităţi exprimată ca în Ecuaţia (2) deoarece evidenţiază ideea de coeziune. Coeziunea se referă la gradul în care componentele modulelor sunt apropiate din punct de vedere conceptual. Distanţa definită în Ecuaţia (2), ilustrează conceptul de coeziune, altfel spus clasele cu disimilarităţi mici sunt coezive, în timp ce clasele cu disimilarităţi mari sunt mai puţin coezive.

## 3. Dezvoltarea unor noi metode pentru rezolvarea problemelor de îmbunătaţire a structurii sistemelor informatice şi identificarea refactorizărilor folosind tehnici de învăţare automată (clustering, reţele cu autoorganizare-SOM, reţele neuronale)

### 3.1 Dezvoltarea de algoritmi bazaţi pe tehnici de învăţare automată pentru identificarea refactorizărilor

Un rezultat semnificativ pe care l-am obţinut în cadrul proiectului este în direcţia *refactorizării adaptive* şi *incrementale*. După cunoştinţele noastre nu există în literatura de specialitate abordări în direcţia refactorizării adaptive şi incrementale, aşa cum a fost abordată de către noi.

Este binecunoscut faptul că îmbunătăţirea structurii sistemelor informatice prin refactorizare este una dintre cele mai importante aspecte de-a lungul evoluţiei sistemelor soft orientate obiect. Problema refactorizării adaptive am definit-o în contextul în care sistemul evoluează, adică noi clase sunt adăugate sistemului pentru a-i adăuga noi funcţionalităţi.

Am dezvoltat o metodă de clustering adaptiv bazată pe o abordare ierarhică aglomerativă, *Hierarchical Adaptive Refactoring (HAR)* [5] precum şi o metodă de clustering incremental bazată pe o metodă de clustering partiţional $k-means$, *Core Based Adaptive Refactoring (CBAR)* [4, 6]. Este vorba despre ajustarea partiţionării stabilite prin aplicarea unui algoritm de clustering (ierarhic aglomerativ: *HAC - Hierarchical Agglomerative Clustering*, sau partiţional *k-means*) înainte de adăugarea noilor clase în sistem, în scopul eficientizării procesului. *HAC* este un algoritm de clustering introdusă anterior de către noi în scopul identificării refactorizărilor care ar îmbunătăţi structura internă a unui sistem informatic.

De asemenea, am introdus un model matematic pentru problema refactorizării adaptive, rezultatele teoretice obţinute fiind validate experimental pe un sistem informatic concret [7]. Metodele adaptive introduse de către noi au identificat mult mai eficient refactorizările necesare pentru a îmbunătăţi structura internă a sistemului informatic extins, fără a scădea acurateţea rezultatelor obţinute.

În cazul în care sistemele informatice evoluează în timp, noi clase sunt adăugate sistemului în scopul îndeplirii unor noi cerinţe funcţionale. Pentru a obţine în aceste condiţii un model de restructurare al sistemului extins, algoritmul de clustering (*HAC* în abordarea noastră), poate fi aplicat de la început, de fiecare dată când mulţimea de clase evoluează. Acest lucru ar presupune ca, de fieare dată când sistemul informatic se modifică, el va fi analizat în totalitate, şi algoritmul $HAC$ este aplicat din nou pentru a obţine o structură internă îmbunătăţită a sistemului. Dar acest lucru poate fi ineficient, mai ales în cazul sistemelor de dimensiuni mari. Soluţia noastră de eficientizare a procesului este introducerea unei metode incrementale care identifică structuri stabile (pe care noi le-am numit nuclee) în cadrul modelului de restructurare al sistemului şi apoi porneşte procesul de clustering de la aceste structuri, adaptate oarecum la evoluţia sistemului (noile clase care au fost adăugate acestuia). Scopul metodei adaptive este de a obţine rezultatele mult mai eficient, reducând complexitatea computaţională a procesului de clustering, fără însă a reduce acurateţea rezultatelor.

În lucrarea [2] am studiat, într-o manieră asemănătoare celei descrisă anterior, problema refactorizării incrementale şi am dezvoltat un algoritm incremental de clustering partiţional numit *Incremental Refactoring Using Seeds* (*IRUS*).

### 3.2 Adaptarea algoritmilor propuşi în vederea identificării refactorizărilor bazate pe aspecte

În lucrarea [1] am introdus un algoritm de clustering ierarhic, *HAC* (Hierarchical Agglomerative Clustering algorithm) pentru identificarea funcţionalităţilor transversale în sisteme informatice existente. Fie $S$ sistemul informatic de analizat constând într-o mulţime de clase $\mathcal{C} = \{c_1, c_2, \ldots, c_s\}$, fiecare clasă conţinând una sau mai multe metode. În procesul de clustering, obiectele care vor fi grupate sunt metodele din sistem, altfel spus, $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$. Scopul este gruparea metodelor în aşa fel încât metodele care aparţine aceleiaşi funcţionalităţi transversale să fie plasate în acelaşi cluster. Folosind metodul vectorial, fiecare metodă din sistem am considerat-o un vector $s$-dimensional $m_i = (m_{i1}, \ldots, m_{is})$, $s$ fiind numărul de clase din $S$. Pentru identificarea metodelor din sistem am considerat două modele vectoriale, iar pentru a ilustra disimilaritatea dintre metode am folosit *distanţa euclidiană*. Algoritmul *HAC* se bazează pe ideea de clustering ierahic aglomerativ care se opreşte când s-a determinat un anumit număr de clusteri. În scopul identificării numărului de clusteri, *HAC* foloseşte o euristică.

De asemenea, în direcţia *aspect mining* ne-am îndreptat cercetările spre dezvoltarea unor măsuri de evaluare a rezultatelor tehnicilor partiţionale de aspect mining [3]. Această direcţie am abordat-o deoarece lipsesc din literatură măsuri pentru evaluarea rezultatelor tehniilor de AM şi a calităţii acestora. De asemenea, am analizat modalitatea în care mşurile originale de evaluare introduse se pot aplica diverselor tehnici de AM existente în literatură.

### 3.3 Implementarea algoritmilor dezvoltaţi şi includerea lor în sistemul multiagent DSSEM

În prototipul sistemului multiagent *DSSEM* pe care l-am dezvoltat în JADE am inclus funcţionalitatea de identificare a aspectelor, pentru a cărei implementare am folosit algoritmul *HAC*.

### 3.4 Evaluarea rezultatelor obţinute şi compararea acestora cu abordări similare existente

Rezultatele obţinute în domeniul identificării funcţionalităţilor transversale au fost evaluate folosind două măsuri de evaluare existente în literatura de specialitate şi o măsură originală de evaluare [1]. Abordarea propusă am comparat-o cu abordări similare existente în *aspect mining* şi am prezentat două studii de caz. De asemenea, abordarea bazată pe clustering pentru identificarea refactorizărilor a fost evaluată pe un studiu de caz open source şi un sistem informatic real, şi comparată cu abordări similare existente în literatura de specialitate. În lucrările originale publicate am evidenţiat avantajele abordărilor propuse de noi faţă de cele deja existente.

## B. Activităţi organizatorice

Activităţile organizatorice au avut ca scop menţinerea unui cadru propice activităţii de cercetare şi îndeplinirii cerinţelor de această natură stipulate în contractul de finanţare. Pagina web a proiectului, dedicată prezentării proiectului, a echipei de cercetare şi a rezultatelor obţinute, http://www.cs.ubbcluj.ro/∼raoss, a fost permanent actualizată. S-au efectuat întâlniri periodice ale echipei de cercetare, în scopul realizării cu succes a activităţilor prevăzute în plan şi diseminarea în cadrul grupului a rezultatelor obţinute.

## C. Diseminarea rezultatelor

Diseminarea rezultatelor ştiinţifice obţinute în anul 2010 în cadrul proiectului şi prezentate în secţiunie anterioare, s-a realizat prin publicarea a 10 articole de specialitate, după cum urmează. **5** publicaţii ISI: **3** în articole cotate ISI - Science Citation Index Expanded (lucrările [1], [2], [3]) şi **2** la conferinţe ISI - Conference Proceedings Citation Index (lucrările [4], [5]) şi **5** articole în reviste indexate BDI (lucrările [6]-[10]). De asemenea, membrii echipei au realizat periodic rapoarte tehnice conţinând rezultatele obţinute în vederea atingerii obiectivelor proiectului.

Menţionăm faptul că factorul de impact cumulat al publicaţiilor în revistele cotate ISI (calculat pe 2009) este 0.616.

De asemenea, alte **5** lucrări au fost trimise la reviste ISI şi sunt în procesul de evaluare.

Ca urmare, criteriul minim de performantă prevăzut (**2** articole ISI şi **3** articole BDI) a fost îndeplinit.

## D. Concluzii

Sintetizăm rezultatele obţinute în cadrul proiectului pe anul 2010 ca fiind următoarele: dezvoltarea sistemului multiagent inteligent DSSEM de suport decizional pentru asistarea dezvoltatorilor în faza de întreţinere şi evoluţie a sistemelor informatice; realizarea unui prototip al sistemului multiagent DSSEM; elaborarea unor modelele teoretice pentru restructurare şi introducerea şabloanelor de proiectare în sisteme informatice existente; şi dezvoltarea unor algoritmi bazaţi pe clustering pentru identificarea refactorizărilor.

Conform celor prezentate anterior, obiectivele planificate pe anul 2010, cât şi activităţile aferente acestora au fost realizate în totalitate, şi desfăşurate conform cu planul de realizare al proiectului. De asemenea, criteriul minim de performantă prevăzut (**2** articole ISI şi **3** articole BDI) a fost îndeplinit.

În concluzie, la încheierea celui de-al doilea an de derulare a proiectului, nu există riscuri de nerealizare sau de întârziere a realizării obiectivelor proiectului. Dimpotrivă, sunt create premizele şi cadrul metodologic pentru începerea activităţilor din următorul an.

# Bibliografie


[1] Czibula, G., Cojocar, G.S, *A hierarchical clustering based approach in aspect mining*, Computing and Informatics, Bratislava, Slovakia, Vol. 29, No. 6, 2010, pp. 881-900 (ISI - Science Citation Index Expanded)

[2] Czibula, G., Czibula, I.G., *Incremental Refactoring Using Seeds*, SIC Journal, Studies in Informatics and Control, Vol. 19, Issue. 3, 2010, pp. 271-284 (ISI - Science Citation Index Expanded)

[3] Czibula, G., Cojocar, G.S., Czibula, I.G., *Evaluation Measures For Partitioning Based Aspect Mining Techniques*, International Journal of Computers, Communications and Control, Proceedings of the International Conference on Computers, Communications and Control, ICCCC 2010, 2010, to be published (ISI - Science Citation Index Expanded)

[4] Czibula, G., Czibula, I.G., *Adaptive Refactoring Using a Core-Based Clustering Approach*, Proceedings of the 9th International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS '10), Cambridge, UK, 2010, pp. 133-138 (ISI - Conference Proceedings Citation Index)

[5] Czibula, I.G., Czibula, G., *Hierarchical Clustering for Adaptive Refactorings Identification*, 2010 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics, AQTR 2010, pp. 99-104 (ISI - Conference Proceedings Citation Index)

[6] Czibula, G., Czibula, I.G., *Clustering Based Adaptive Refactoring*, Wseas Transactions on Information Science and Applications, Issue 3, Volume 7, 2010, pp. 391-400 (indexed Scopus)

[7] Czibula, I.G., Czibula, G., *Adaptive restructuring of object-oriented software systems*, Studia Universitatis "Babes-Bolyai", Informatica, LV(2), 2010, pp. 1-12 (indexed MathSciNet)

[8] Czibula, I.G., Czibula, G., *On converting software systems to object oriented architectures*, BRAIN Journal, Special Issue on Complexity in Sciences and Artifical Intelligence, Vol. 1, 2010, pp. 12-17 (indexed EBSCO, Ulrichs)

[9] Czibula, G., Czibula, I.G., Pintea, C.M, *A reinforcement learning approach for solving the matrix bandwidth minimization problem*, Studia Universitatis "Babes-Bolyai", Informatica, LV(4), 2010, pp. (indexed MathSciNet)

[10] Czibula, I.G., Czibula, G., *Unsupervised transformation of procedural programs to object-oriented design*, Acta Universitatis Apulensis, 2010, to be published (indexed Mathematical Reviews)

[11] http://www.fipa.org/

[12] http://sourceforge.net/projects/ingenias/


**Cluj-Napoca, 15 noiembrie 2010**

**Director de proiect,**
**Prof. dr. CZIBULA GABRIELA**